February 1994

A Two-Dimensional Hierarchy for Parallel Rewriting Systems

Owen Rambow
University of Pennsylvania

Giorgio Satta
Universita di Venezia

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-94-02.

A Two-Dimensional Hierarchy for Parallel Rewriting Systems

Abstract
The class of parallel rewriting systems is considered in this work, and the interaction between two complexity measures, that in the literature have been called synchronous parallelism and independent parallelism, is investigated. It is shown that, when the degree of synchronous parallelism is bounded by some constant greater than one, the degree of independent parallelism induces an infinite non-collapsing hierarchy within the family of generated languages. The result is obtained using an original characterization of parallel rewriting systems.

Our result combines with other well known properties of synchronous parallelism to reveal the existence of a two-dimensional hierarchy for the family of languages generated by so called finite copying parallel rewriting systems. This gives a new picture of many formalisms in this class. Other language-theoretic properties of parallel rewriting systems are proved in this work, that together with our main result provide an answer to some questions that were left open in the literature.

Comments
University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-94-02.

The Institute For Research In Cognitive Science

A Two-Dimensional Hierarchy for Parallel Rewriting Systems

by

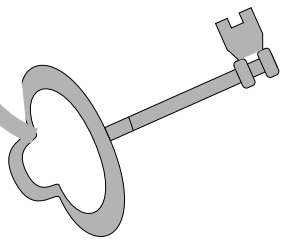
Owen Rambow
University of Pennsylvania

Giorgio Satta
Università di Venezia

University of Pennsylvania
Philadelphia, PA 19104-6228

February 1994

Site of the NSF Science and Technology Center for Research in Cognitive Science



University of Pennsylvania
Founded by Benjamin Franklin in 1740

IRCS Report 94-02

A Two-Dimensional Hierarchy for Parallel Rewriting Systems*

Owen Rambow(†)
Giorgio Satta(‡)

(†) Department of CIS
University of Pennsylvania
3401 Walnut Street, Suite 400C, Philadelphia, PA 19104
rambow@linc.cis.upenn.edu

(‡) Università di Venezia
Scienze dell'Informazione
via Torino, 155 30172 Mestre – Venezia, Italy
satta@moo.dsi.unive.it

Abstract

The class of parallel rewriting systems is considered in this work, and the interaction between two complexity measures, that in the literature have been called synchronous parallelism and independent parallelism, is investigated. It is shown that, when the degree of synchronous parallelism is bounded by some constant greater than one, the degree of independent parallelism induces an infinite non-collapsing hierarchy within the family of generated languages. The result is obtained using an original characterization of parallel rewriting systems.

Our result combines with other well known properties of synchronous parallelism to reveal the existence of a two-dimensional hierarchy for the family of languages generated by so called finite copying parallel

*We are grateful to Joost Engelfriet and Ryuichi Nakanisi for helpful discussion on topics related to this paper. This research was conducted while the second author was a post-doctoral fellow at the Institute for Research in Cognitive Science at the University of Pennsylvania. The research was sponsored by the following grants: ARO DAAL 03-89-C-0031; DARPA N00014-90-J-1863; NSF IRI 90-16592; and Ben Franklin 91S.3078C-1.

rewriting systems. This gives a new picture of many formalisms in this class. Other language-theoretic properties of parallel rewriting systems are proved in this work, that together with our main result provide an answer to some questions that were left open in the literature.

1 Introduction

Since the early seventies, many rewriting systems have been presented in the formal language literature that extend the generative power of the class of context-free grammars. A family of these formalisms, called parallel rewriting systems, has been extensively investigated; the reader is referred to [Engelfriet *et al.*, 1980] for an excellent review of many important results about this family.

Two different kinds of parallelism are realized in parallel rewriting systems, that in [Engelfriet *et al.*, 1980, p.151] have been called the *synchronized* parallelism and the *independent* parallelism. The synchronized parallelism allows derivations of substrings to proceed in a synchronous way, that is a sentence in the generated language may include substrings that have been obtained by a common underlying derivation process. The independent parallelism reflects the capability of the system to instantiate independent derivation processes that are combined together to form the generated string. Context-free grammars are the canonical example of a system with only independent parallelism, while the ET0L systems of [Rozenberg, 1973] are an example of parallel rewriting systems with only synchronized parallelism. Examples of rewriting systems using both kinds of parallelism include the generalized syntax-directed translation (GSDT) of [Aho and Ullman, 1971] and the top-down tree-to-string transducers (yT) of [Engelfriet *et al.*, 1980] (A transducer can be regarded as a controlled generative device.) Both classes generate the same languages. An interesting correspondence between the class of parallel rewriting systems and the class of two-ways machines has been established in [Engelfriet *et al.*, 1980], where equivalence in generative power is shown using a generalized model called checking tree-pushdown transducer.

If we restrict the synchronized parallelism to a finite degree, that is if we allow only a finite number of subderivations to be synchronized in a given grammar, we obtain a subfamily of parallel rewriting systems that includes the so called finite copying top-down tree-to-string transducers (yT_{fc}) of [En-

gelfriet *et al.*, 1980], the string generating context-free hypergraph grammars (CFHG) of [Bauderon and Courcelle, 1987], the multiple context-free grammars (MCFG) of [Kasami *et al.*, 1987] and [Seki *et al.*, 1991] and the string-based linear context-free rewriting systems (LCFRS) of [Vijay-Shanker *et al.*, 1987; Joshi *et al.*, 1991]. All these rewriting systems are weakly equivalent, as shown in [Engelfriet and Heyker, 1991] and in [Weir, 1992]. As far as the correspondence with two-ways machines is concerned, the family of parallel rewriting systems with finite synchronous parallelism generates the same languages as the class of deterministic tree-walking transducers (DTWT) of [Aho and Ullman, 1971].

At the same time, the size of the longest production in a given parallel rewriting system always imposes a finite bound on the number of independent derivation processes that can be interleaved. Hence the degree of independent parallelism is always bounded by some constant in parallel rewriting systems. In what follows, we will regard the two kinds of parallelism introduced above as complexity measures.

Independent investigations of different formalisms in the family of parallel rewriting systems with finite degree of synchronized parallelism have shown that this measure of complexity establishes an infinite, non-collapsing hierarchy in the generated languages; that is, by increasing the degree of synchronized parallelism we gain additional generative power (see for instance [Engelfriet *et al.*, 1980], [Habel and Kreowsky, 1987] and [Seki *et al.*, 1991]). As an example, if the degree of synchronized parallelism is bounded by an integer $f \geq 1$, a grammar can "count" up to $2f$, but cannot generate the language $L = \{a_1^n a_2^n \cdots a_{2f+1}^n \mid n \geq 0\}$. Whether the second complexity measure, that is independent parallelism or maximum production size, induces a corresponding infinite non-collapsing hierarchy in the generated languages was not known to date. The major contribution of this paper is the solution of this problem. We show that, within parallel rewriting systems with synchronized parallelism bounded by $f \geq 2$, the degree of independent parallelism induces an infinite non-collapsing hierarchy for the generated languages.

We study how the two complexity measures interact and show the existence of a two-dimensional hierarchy for the class of parallel rewriting systems with finitely bounded degree of synchronous parallelism (so called finite copying parallel rewriting systems). We investigate language theoretic properties of the members of such a hierarchy, and solve in the negative a question left open in [Engelfriet *et al.*, 1980], about whether parallel rewriting sys-

tems with degree of synchronous parallelism bounded by a fixed constant constitute a full principal AFL. When the degree of synchronous parallelism is bounded by $f = 1$, that is when synchronous parallelism is inhibited, the above rewriting systems can be cast in a normal form defined by some bound on the size of the productions, that is some bound on the degree of independent parallelism. (As already mentioned, in this case the generated languages are exactly the context-free languages and the above fact is related to the existence of two-normal forms for context-free grammars.) Our result shows that, when $f \geq 2$, such normal forms are not admitted. This solves a question left open in [Aho and Ullman, 1971] and has interesting consequences for the design of algorithms for the recognition problem for these rewriting systems.

In a sense, our result is a generalization of a result in [Aho and Ullman, 1969], concerning non-simple syntax-directed translation schemata, a restricted kind of parallel rewriting systems with degree of synchronous parallelism bounded by $f = 2$. There, the existence of an infinite non-collapsing hierarchy induced by the degree of independent parallelism is shown for such systems. We generalize this formalism by introducing a new class of rewriting systems called local unordered scattered-context grammar (LUSCG) which has the same generative power as parallel rewriting systems with finitely bounded degree of synchronous parallelism. The definition of LUSCG is based on a rewriting restriction, called locality, that turns out to provide an exact characterization of finite copying parallel rewriting systems. We then show our result by working on LUSCG in such a way that we can then transfer our results to all other formalisms mentioned above. The choice of LUSCG renders the proof of our result more intuitive, due to the intrinsic parallelism of these systems.

This paper is organized in the following way. In Section 2 we introduce local unordered scattered context grammars and define two parameters for this class that are related to synchronous and independent parallelism. In Section 3 we show that the degree of independent parallelism induces an infinite hierarchy when the degree of synchronous parallelism is bounded by a constant, and in Section 4 we prove some language theoretic properties for members of such a hierarchy. In Section 5 we use an equivalence result reported in Appendix A to show how all our results can be transferred to other formalisms in the class of parallel rewriting systems; in this way we show the existence of a two-dimensional hierarchy for these systems. Finally, in Section 6 we discuss some other consequences of the presented results.

2 Definitions

In this section we introduce a new class of parallel rewriting systems, which we will call local unordered scattered context grammar, and show how valid derivations in these systems can be represented by means of trees generated by context-free grammars. We then define two (independent) parameters for this class, called fan-out and rank. These two parameters are related to synchronous and independent parallelism, and their mutual interaction will be investigated throughout this paper. In what follows we will use standard notational conventions. For an alphabet V, we denote by V^* the set of all finite strings over V, V^+ the set of non-empty ones. Let $a \in V$ and $w \in V^*$; $\#_a(w)$ denotes the number of occurrences of a in w. As usual, for a class $\mathcal{C}$ of generative devices, $\mathcal{L}(\mathcal{C})$ denotes the class of all languages generated by $\mathcal{C}$.

A class of rewriting systems called scattered context grammars was introduced by [Greibach and Hopcroft, 1969]; an unordered version was proposed by [Milgram and Rosenfeld, 1971] and [Mayer, 1972]. The following definition is based on [Salomaa, 1973, p.259] and [Dassow and Păun, 1989, p.135].

Definition 1 *An* unordered scattered context grammar *(*USCG *for short) is a quadruple $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, P, S)$ where V_{N}, V_{T} are finite, disjoint sets of non-terminal and terminal symbols respectively, $S \in V_{\mathrm{N}}$ is the start symbol and P is a finite set of productions having the form $(A_1, \ldots, A_n) \to (\alpha_1, \ldots, \alpha_n)$, where $n \geq 1$, $A_i \in V_{\mathrm{N}}$, $\alpha_i \in (V_{\mathrm{N}} \cup V_{\mathrm{T}})^*$, $1 \leq i \leq n$.*

We write $\gamma \Rightarrow_G \delta$ whenever there exist $p = (A_1, \ldots, A_n) \to (\alpha_1, \ldots, \alpha_n) \in P$ and an arbitrary permutation π of $\{1, \ldots, n\}$ such that

$$\gamma = \gamma_0 A_{\pi(1)} \gamma_1 A_{\pi(2)} \cdots \gamma_{n-1} A_{\pi(n)} \gamma_n,$$

$$\delta = \gamma_0 \alpha_{\pi(1)} \gamma_1 \alpha_{\pi(2)} \cdots \gamma_{n-1} \alpha_{\pi(n)} \gamma_n,$$

where $\gamma_i \in (V_{\mathrm{N}} \cup V_{\mathrm{T}})^*$, $0 \leq i \leq n$.

The class of all unordered scattered context grammars is also denoted USCG. USCGs are known to be weakly equivalent to several other regulated rewriting systems, including context-free matrix grammars and state grammars. The reader is referred to [Dassow and Păun, 1989] for details.

We now introduce a restriction on the derivation relation for USCG which we will call *locality*. Informally, locality forces each production to rewrite only symbols which were previously introduced together in a single step of the derivation. As a result, we have that in a local rewriting system the set of all derivations can be characterized by a recognizable set in the sense of [Thatcher, 1973], i.e., each derivation can be represented by a tree generated by a (fixed) context-free grammar. The notion of locality was first discussed in [Weir, 1988] and can be used to characterize the class of finite copying parallel rewriting systems, as it will be discussed in Section 5.

In what follows, strings $\gamma \in (V_N \cup V_T)^*$ will be viewed as sequences of symbols. An equivalence relation I_γ is said to be *associated with* γ if I_γ is defined on the set of elements of γ that are instances of symbols in V_N. We introduce associated equivalence relations in the definition of the derive relation for USCG to define a new class of rewriting systems. (We overload symbol $\Rightarrow_G$.)

Definition 2 *A* local unordered scattered context grammar *(*LUSCG *for short) is an unordered scattered context grammar G associated with a binary relation $\Rightarrow_G$ defined over pairs consisting of a string in $(V_N \cup V_T)^*$ and an associated equivalence relation. We write $(\gamma, I_\gamma) \Rightarrow_G (\delta, I_\delta)$ if and only if:*

(i) *δ is obtained from γ by using a production $p \in P$ to rewrite elements of γ that are equivalent in I_γ, and*

(ii) *$I_\delta = I'_\gamma \cup I$, where I'_γ makes equivalent all and only those instances of nonterminals in δ that have not been introduced by p and that correspond to instances that were equivalent in I_γ, and I makes equivalent all and only the instances of nonterminal elements of δ introduced by p.*

The class of all local unordered scattered context grammars is also denoted LUSCG. The relationship between the non-local and local versions of USCG will be discussed in Section 6.

As a convention, given a string of the form $\gamma_0 A_1 \gamma_1 \cdots \gamma_{n-1} A_n \gamma_n$, $n \geq 1$, we denote with $I^{\langle A_1,\ldots,A_n\rangle}$ any associated equivalence relation in which the indicated instances of nonterminals $A_1, \ldots, A_n$ are equivalent. We introduce additional notation to be used in the following. If $p : (A_1, \ldots, A_n) \rightarrow (\alpha_1, \ldots, \alpha_n)$ belongs to P, we say that $(A_1, \ldots, A_n)$ is the *left-hand tuple* of p and $(\alpha_1, \ldots, \alpha_n)$ is the *right-hand tuple* of p. Relation $\Rightarrow_G$ will sometime

be written $\stackrel{p}{\Rightarrow}_G$ to indicate that production p was used in the rewriting. In order to represent derivations in G, we use the reflexive and transitive closure of $\Rightarrow_G$, written $\stackrel{*}{\Rightarrow}_G$.

The *reduced form* for an LUSCG G is the LUSCG derived from G by eliminating useless productions, i.e., productions that can never be used in a terminating derivation. It can easily be shown that for each LUSCG G, there is a reduced form grammar. (The construction is analogous to the case of context-free grammars.) In this paper, we will always assume that a grammar is in reduced form.

$$
\begin{array}{rll}
G_L = & (V_{\mathrm{N}}, V_{\mathrm{T}}, P, S); & \\
V_{\mathrm{N}} = & \{S, A, B\} & \\
V_{\mathrm{T}} = & \{[,]\} & \\
P = & \{\mathrm{p}_1\colon & (S) \longrightarrow (AA), \\
 & \mathrm{p}_2\colon & (A, A) \longrightarrow ([A], [A]), \\
 & \mathrm{p}_3\colon & (A, A) \longrightarrow (AB, AB), \\
 & \mathrm{p}_4\colon & (B, B) \longrightarrow (A, A), \\
 & \mathrm{p}_5\colon & (A, A) \longrightarrow ([\,], [\,])\}
\end{array}
$$

Figure 1: An LUSCG for language $L = \{ww | w \in D_1\}$.

Example 1 Let D_1 be the Dyck language of strings of properly balanced parentheses and let $L = \{ww | w \in D_1\}$. Language L can be derived by the LUSCG G_L specified in Figure 1. ■

As already mentioned, the locality restriction makes it possible to represent the underlying structure of a derivation by means of a recognizable tree. The following definition specifies how this can be done. Let $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, P, S)$ be a LUSCG. Define $P^{(0)} = \{p \mid p \in P$, there are no nonterminals in the right-hand tuple of $p\}$ and $P^{(1)} = P - P^{(0)}$. Without loss of generality, we assume that p_S is the unique production in P that rewrites S and $p_S \in P^{(1)}$.

Definition 3 *The* derivation grammar of G, *denoted* $\mathrm{der}(G)$, *is the context-free grammar* $(P^{(1)}, P^{(0)}, \Pi, p_S)$ *where* $P^{(1)}$ *and* $P^{(0)}$ *are the set of nonterminal and terminal symbols respectively,* p_S *is the initial symbol and* Π *contains all and only productions of the form* $p \to p_1 \cdots p_n$, *where* $p, p_1, \ldots, p_n \in P$ *and* $n \geq 1$, *such that* $p_1, \ldots, p_n$ *together rewrite all the nonterminal symbols*

of G introduced by the right-hand tuple of p.

We remark that Definition 3 assumes a canonical ordering of the productions, so that two productions of $\mathrm{der}(G)$ cannot differ only in the order of the right-hand symbols. Clearly, every derivation in G corresponds to a unique derivation in $\mathrm{der}(G)$.

Example 1 (continued) The derivation grammar of G_L is given in Figure 2. ■

We conclude present section with the definition of two parameters associated with grammars in the class LUSCG. In the next sections these parameters will be considered as complexity measures and their interaction will be investigated.

$$
\begin{aligned}
\mathrm{der}(G_L) &= (P^{(1)}, P^{(0)}, \Pi, p_1);\\
P^{(1)} &= \{p_1, p_2, p_3, p_4\}\\
P^{(0)} &= \{p_5\}\\
\Pi &= \{\ p_1 \longrightarrow p_2,\quad p_1 \longrightarrow p_3,\quad p_1 \longrightarrow p_5,\\
&\qquad p_2 \longrightarrow p_2,\quad p_2 \longrightarrow p_3,\quad p_2 \longrightarrow p_5,\\
&\qquad p_3 \longrightarrow p_2p_4,\quad p_3 \longrightarrow p_3p_4,\quad p_3 \longrightarrow p_4p_5,\\
&\qquad p_4 \longrightarrow p_2,\quad p_4 \longrightarrow p_3,\quad p_4 \longrightarrow p_5\ \}
\end{aligned}
$$

Figure 2: Derivation grammar for G_L.

Definition 4 *Let $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, P, S)$ be a* LUSCG*, $p \in P$, and let $\mathrm{der}(G) = (P^{(1)}, P^{(0)}, \Pi, p_S)$ be the derivation grammar of G. The* fan-out *of production p, written $\varphi(p)$, is the length of its tuples. The fan-out of G is defined as $\varphi(G) = \max_{p \in P} \varphi(p)$. The* rank *of production p, written $\rho(p)$, is defined as $\rho(p) = \max_{(p \to \alpha) \in \Pi} |\alpha|$. The rank of G is defined as $\rho(G) = \max_{p \in P} \rho(p)$.*

For integers $f \geq 1$ and $r \geq 0$, LUSCG(f) will denote the class of all LUSCG having fan-out bounded by f and r-LUSCG will denote the class of all LUSCG with rank bounded by r; r-LUSCG(f) will denote the intersection of the two. We remark that a grammar $G \in$ LUSCG must have fan-out and rank at least as great as those of its reduced form grammar; put differently, the process of eliminating useless productions can only decrease, but never increase, the fan-out and rank of a grammar.

3 A rank hierarchy

This section presents the main result of the paper. We show that the rank parameter defines an infinite (non-collapsing) hierarchy within each class LUSCG(f), $f \geq 2$. The technique we have adopted has been inspired by a technique used in [Aho and Ullman, 1969] to prove the existence of an infinite hierarchy induced by the rank in non-simple syntax-directed translation schemata (SDTS). Indeed, the definition we have given for the derive relation for LUSCG can be seen as a generalization of the definition of derivation in SDTS.

Let $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, P, S)$ be a (reduced) LUSCG. We first introduce some notions that describe productions of G in terms of derivations in which they can participate.

Definition 5 *A production p in P with left-hand tuple $(A_1, \ldots, A_t), t \geq 1$,* covers *terminal symbol $a \in V_{\mathrm{T}}$ if and only if for any integer $d \geq 1$ there exists a derivation η such that the following conditions are satisfied:*

(i) *η starts with p and has the form*

$$(\gamma_0 A_{\pi(1)} \gamma_1 \ldots \gamma_{t-1} A_{\pi(t)} \gamma_t, I^{\langle A_1, \ldots, A_t \rangle}) \stackrel{*}{\Rightarrow}_G (\gamma_0 v_1 \gamma_1 \ldots \gamma_{t-1} v_t \gamma_t, I),$$

for some equivalence relation I, where $\gamma_i \in (V_{\mathrm{N}} \cup V_{\mathrm{T}})^*$, $0 \leq i \leq t$, and $v_i \in V_{\mathrm{T}}^*$, $1 \leq i \leq t$;

(ii) *string $v_1 \cdots v_t$ includes more than d instances of a.*

In the following we will use symbol $\lhd$ to denote the covering relation, and we will take $p \lhd \{a_1, \ldots, a_k\}$ to mean that $p \lhd a_i$ for each i, $1 \leq i \leq k$. Furthermore, we will write $A \lhd a$ if A is a nonterminal in the left-hand tuple of some production p, which is understood from the context, and $p \lhd a$ in such a way that an unbounded number of instances of a are included in the substring derived by A.

Let $p : (A_1, \ldots, A_t) \to (\alpha_1, \ldots, \alpha_t), t \geq 1$, be a production in P, and let $a \in V_{\mathrm{T}}$. Consider derivations in G of the form

$$\begin{aligned} (S, I^{\langle S \rangle}) &\stackrel{*}{\Rightarrow}_G (u_0 A_1 u_1 \cdots u_{t-1} A_t u_t, I_1) \\ &\stackrel{p}{\Rightarrow}_G (u_0 \alpha_1 u_1 \cdots u_{t-1} \alpha_t u_t, I_2) \\ &\stackrel{*}{\Rightarrow}_G (u_0 v_1 u_1 \cdots v_{t-1} w_t v_t, I_3), \end{aligned} \tag{1}$$

where $u_i, v_j \in (V_T)^*$. If p does not cover a, it follows that there exists a constant $M_{p,a} \geq 0$ such that, for any derivation of the form in (1), the number of instances of a in $v_1 v_2 \cdots v_t$ is bounded by $M_{p,a}$. If $p \lhd a$, we assume $M_{p,a} = -1$. We define M_G to be the maximum among all $M_{p,a}$, $p \in P$ and $a \in V_T$. Furthermore, let $L_M(G) = \{w \mid w \in L(G), \#_a(w) > M_G \text{ for every } a \in V_T\}$. A production p of G is called *productive* if $L_M(G) \neq \emptyset$ and p is used in some derivation for some sentence in $L_M(G)$. This property will be used to exclude productions of G that can generate only uninteresting sets of strings.

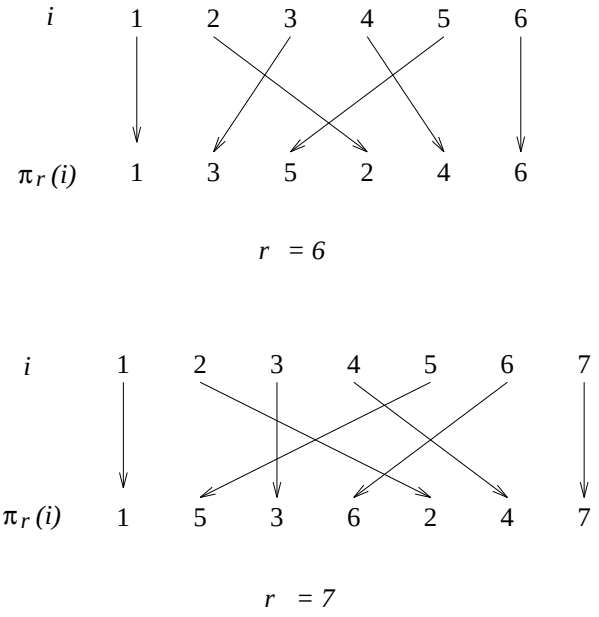


Figure 3: The permutation π_r for $r = 6$ and $r = 7$.

We first prove a separation result for classes r-LUSCG(f), $r \geq 3$. In order to do so, we define a particular family of languages to which we will henceforth restrict our attention.

Definition 6 *Let r and f be two integers, $r, f \geq 1$. Let also $V_T^{(r,f)} = \{a_{i,j} \mid 1 \leq i \leq r, 1 \leq j \leq f\}$ and π_r a permutation of $\{1, 2, \ldots, r\}$ defined as*

follows. If r is even:

$$\pi_r(i) \;=\; \begin{cases} 2i-1, & i \in \{1,\ldots,r/2\}; \\ 2i-r, & i \in \{r/2+1,\ldots,r\}. \end{cases}$$

If r is odd:

$$\pi_r(i) \;=\; \begin{cases} i, & i \in \{1,r\}; \\ r-1, & i = (r+1)/2; \\ r-2(i-1), & i \in \{2,\ldots,(r+1)/2-1\}; \\ 2i-r-1, & i \in \{(r+1)/2+1,\ldots,r-1\}. \end{cases}$$

Language $L_{r,f}$ is specified as follows:

$$\begin{aligned} L_{r,f} \;=\; & \{w_1 w_2 \cdots w_f \mid w_1 = a_{1,1}^{i_1} \cdots a_{r,1}^{i_r},\ w_h = a_{\pi_r(1),h}^{i_{\pi_r(1)}} \cdots a_{\pi_r(r),h}^{i_{\pi_r(r)}}, \\ & 2 \le h \le f, i_j \ge 1,\ 1 \le j \le r\}. \end{aligned}$$

The effect of π_r in the cases $r = 6$ and $r = 7$ is shown for illustrative purposes in Figure 3.

In the following we will call *segment* each substring w_h in the definition of a string in $L_{r,f}$. We will also use $\overline{a_s}$ to denote the set $\{a_{s,1}, a_{s,2} \ldots, a_{s,f}\}$, which we will refer to as a *terminal group* for $L_{r,f}$. The set of all terminal groups for $L_{r,f}$ will be denoted $\mathcal{V}^{(r,f)}$ and, for $r \ge 3$, set $\{\overline{a_i} \mid 2 \le i \le r-1\}$ will be denoted $\mathcal{B}^{(r,f)}$. The definition of $\lhd$ extends to a set of sets in the obvious way. We now relate previous definitions by means of an example.

Example 2 Language $L_{r,f}$ can be derived by a grammar in $(r-2)$-LUSCG(f), for $f \ge 1$ and $r \ge 4$: such a grammar is defined in Figure 4. We have $\rho(p_1) = \rho(p_2) = 2$, $\rho(p_3) = r-2$ and $\rho(p_{3+j}) = 1$, $\rho(p_{3+r+j}) = 0$ for $1 \le j \le r$; the rank of p_3 determines the rank of G. Observe that p_3 covers $\mathcal{B}^{(r,f)}$. ■

In Lemma 4 below we will show a basic fact about our family of languages, namely we prove that, for any $r \ge 6$ and $f \ge 2$, any grammar in LUSCG that derives $L_{r,f}$ cannot have a production that covers more than one, but fewer than $r-3$, terminal groups in $\mathcal{B}^{(r,f)}$. To do this, we need first some intermediate results. In the following discussion, we will be referring to an implicit LUSCG generating $L_{r,f}$; hence, for example, whenever we mention a symbol $a_{s,q}$, the ranges of s and q are implicitly stated.

The first lemma shows that for languages $L_{r,f}$, the properties of covering $a_{s,q}$ and of covering $\overline{a_s}$ cannot be distinguished.

$$
\begin{array}{lcl}
G & = & (V_{\mathrm{N}}, V_{\mathrm{T}}^{(r,f)}, S, P); \\
P & = & \{p_i \quad | \quad 1 \le i \le 3 + 2r\}; \\
V_{\mathrm{N}} & = & \{S\} \cup \{Q_j, R_j \quad | \quad 1 \le j \le f\} \cup \\
 & & \qquad \{A_{i,j} \quad | \quad 1 \le i \le r, 1 \le j \le f\};
\end{array}
$$

$$
\begin{array}{lcl}
p_1 & : & (S) \to (A_{1,1} Q_1 \cdots A_{1,f} Q_f); \\
p_2 & : & (Q_1, \ldots, Q_f) \to (R_1 A_{r,1}, \ldots, R_f A_{r,f}); \\
p_3 & : & (R_1, \ldots, R_f) \to (\alpha^{(1)}, \ldots, \alpha^{(f)}), \quad \alpha^{(1)} = A_{2,1} A_{3,1} \cdots A_{r-1,1}, \\
 & & \qquad \alpha^{(j)} = A_{\pi_r(2),j} A_{\pi_r(3),j} \cdots A_{\pi_r(r-1),j}, \quad 2 \le j \le f; \\
p_{3+j} & : & (A_{j,1}, \ldots, A_{j,f}) \to (a_{j,1} A_{j,1}, \ldots, a_{j,f} A_{j,f}), \quad 1 \le j \le r; \\
p_{3+r+j} & : & (A_{j,1}, \ldots, A_{j,f}) \to (a_{j,1}, \ldots, a_{j,f}) \quad 1 \le j \le r;
\end{array}
$$

Figure 4: A $(r-2)$-LUSCG(f) grammar for $L_{r,f}$.

Lemma 1 *If a production p covers some $a_{s,q}$, then p covers $\overline{a_s}$.*

Proof. If $f = 1$, the statement trivially holds. For $f > 1$, assume there exists $q' \neq q$ such that p does not cover $a_{s,q'}$. Consider a derivation of the form

$$
\begin{array}{rcl}
(S, I^{\langle S \rangle}) & \stackrel{*}{\Rightarrow}_G & (u_0 A_1 u_1 \cdots u_{t-1} A_t u_t, I_1) \\
 & \stackrel{p}{\Rightarrow}_G & (u_0 \alpha_1 u_1 \cdots u_{t-1} \alpha_t u_t, I_2) \\
 & \stackrel{*}{\Rightarrow}_G & (u_0 v_1 u_1 \cdots u_{t-1} v_t u_t, I_3),
\end{array} \tag{2}
$$

where $u_i, v_j \in (V_{\mathrm{T}}^{(r,f)})^*$, and $t \ge 1$. (Note that such a derivation exists since we assume the grammar to be reduced.) Let m be the number of instances of $a_{s,q'}$ in $u_0 u_1 \cdots u_t$. Since p covers $a_{s,q}$, we can derive a second string in $L_{r,f}$ of the form $u_0 x_1 u_1 \cdots u_{t-1} x_t u_t$, such that the number of instances of $a_{s,q}$ in string $x_1 x_2, \cdots x_t$ exceeds $M_G + m$. Since $a_{s,q'}$ is not covered by p, the number of instances of $a_{s,q'}$ in $x_1 x_2 \cdots x_t$ is bounded by M_G. Then the number of instances of $a_{s,q}$ differs from that of $a_{s,q'}$, contradicting the definition of $L_{r,f}$. ∎

Next we show that whenever a nonterminal A in the left-hand tuple of a productive production p covers two different symbols $a_{s,q}$ and $a_{s,q'}$, that is, two symbols belonging to the same terminal group but to different segments, then p covers the whole of $V_{\mathrm{T}}^{(r,f)}$. We prove the result in two steps.

Let a and b be two symbols in $V_{\mathrm{T}}^{(r,f)}$. Observe that the set of all terminal symbols occurring between a and b (including a and b) is the same for all

strings in $L_{r,f}$. We will call such a set the *in-between set* of a and b. From the definition of $L_{r,f}$ it follows that if a' is in the in-between set of a and b, $a' \neq a$ and $a' \neq b$, then in any string in $L_{r,f}$, all instances of a' occur between instances of a and instances of b.

Lemma 2 *Let $r \geq 2$ and let p be a productive production such that a nonterminal symbol A in the left-hand tuple of p covers (by means of p) set $\{a, b\} \subseteq V_{\mathrm{T}}^{(r,f)}$. Then p covers the in-between set of a and b.*

Proof. If the in-between set of a and b is $\{a, b\}$, the lemma holds trivially. Let a' be in the in-between set of a and b, $a' \notin \{a, b\}$, and suppose that p cannot generate more than M_G instances of a'. Since p is productive, there exists a sentential derivation using p that derives a string in $L_{r,f}$ which has more than M_G occurrences of a'. Such a derivation can be represented as

$$\begin{aligned} (S, I^{\langle S \rangle}) &\overset{*}{\Rightarrow}_G (u_0 A_1 u_1 \cdots u_{k-1} A_k u_k \cdots u_{t-1} A_t u_t, I_1) \\ &\overset{p}{\Rightarrow}_G (u_0 \alpha_1 u_1 \cdots u_{k-1} \alpha_k u_k \cdots u_{t-1} \alpha_t u_t, I_2) \\ &\overset{*}{\Rightarrow}_G (u_0 v_1 u_1 \cdots u_{k-1} v_k u_k \cdots u_{t-1} v_t u_t, I_3), \end{aligned} \tag{3}$$

where $A = A_k$, $u_i, v_j \in (V_{\mathrm{T}}^{(r,f)})^*$, and $t \geq 1$. Since $p \not\lhd a'$ by assumption, there must be instances of a' in $u_0 u_1 \cdots u_t$. A_k generates a and b by means of p; we can therefore derive a second string in $L_{r,f}$ having the form $w = u_0 x_1 u_1 \cdots u_{k-1} x_k u_k \cdots u_{t-1} x_t u_t$, such that x_k contains instances of a and b. But then we have instances of a' in w not occurring between a and b: this contradicts the definition of $L_{r,f}$. ■

We can now prove the previously mentioned result.

Lemma 3 *Let $f \geq 2$ and $r \geq 2$. If a nonterminal A from the left-hand tuple of a productive production p covers both $a_{s,q}$ and $a_{s,q'}$ in $V_{\mathrm{T}}^{(r,f)}$, for some s and $q < q'$, then p covers $\mathcal{V}^{(r,f)}$.*

Proof. Since $a_{r,q}$ and $a_{1,q+1}$ are in the in-between set of $a_{s,q}$ and $a_{s,q'}$, p must cover both $a_{r,q}$ and $a_{1,q+1}$ by Lemma 2. By Lemma 1, p must cover $\overline{a_1}$ and $\overline{a_r}$. If one nonterminal B from the left-hand tuple of p covers more than one member of $\overline{a_1}$, say $a_{1,u}$ and $a_{1,u'}$, then by Lemma 2 we have $p \lhd a_{j,u}$ for $1 \leq j \leq r$, and we are done by Lemma 1. Suppose instead that each nonterminal from the left-hand tuple of p covers exactly one member of $\overline{a_1}$. Let B be the nonterminal that covers $a_{q,1}$. If B covers any member of $\overline{a_r}$,

then it must cover $a_{r,1}$. By Lemmas 2 and 1, we are done. Suppose instead that B does not cover any member of $\overline{a_r}$. Then there must be a nonterminal from the left-hand tuple of p which covers two members of $\overline{a_r}$, say $a_{r,u}$ and $a_{r,u'}$. Then by Lemma 2, p covers $a_{j,u'}$ for $1 \leq j \leq r$, and we are done by Lemma 1. ■

We now use previous results to derive a basic property of productive productions in G that will be used to show the major result.

Lemma 4 *Let $f \geq 2$ and $r \geq 6$. If a productive production p covers more than one terminal group in $\mathcal{B}^{(r,f)}$, then p covers $\mathcal{B}^{(r,f)}$.*

Proof. Assume that $(A_1, \ldots, A_t)$, $t \geq 1$, is the left-hand tuple of p. First we show that under the above hypotheses, if $p \lhd \{\overline{a_s}, \overline{a_{s'}}\}$ for $\overline{a_s}, \overline{a_{s'}} \in \mathcal{B}^{(r,f)}$, $s < s'$, then the only interesting case for us is $t = f$ and $A_i \lhd \{a_{s,i}, a_{s',i}\}$ for $1 \leq i \leq f$. Since p is productive, if any nonterminal A in the left-hand tuple of p covers $\{a_{s,q}, a_{s,q'}\}$, $q \neq q'$, then by Lemma 3 p covers all of the terminal groups in $\mathcal{V}^{(r,f)}$. The remaining possibility is that $t = f$ and for all i, $1 \leq i \leq f$, A_i covers exactly one terminal in $\overline{a_s}$ and exactly one terminal in $\overline{a_{s'}}$. W.l.o.g. we may assume that $A_i \lhd a_{s,i}$, $1 \leq i \leq f$. From the definition of $L_{r,f}$, it follows that $A_i \lhd a_{s',i}$, $1 \leq i \leq f$. In the following, we will therefore deal only with the case $A_i \lhd \{a_{s,i}, a_{s',i}\}$ for $1 \leq i \leq f$.

Since A_1 covers both $a_{s,1}$ and $a_{s',1}$ by means of p and since p is productive, by Lemma 2 we conclude that p must also cover $a_{s+1,1}$, and hence $\overline{a_{s+1}}$ by Lemma 1. Again we restrict our attention to the only interesting case in which $A_i \lhd \{a_{s,i}, a_{s+1,i}\}$, $1 \leq i \leq f$. By investigating the case $i = 2$, we now show that $p \lhd \{a_{r-1,2}, a_{2,2}\}$. We distinguish three cases.
Case 1: r is even. It can be seen from the definition of $L_{r,f}$ that $a_{r-1,2}$ and $a_{2,2}$ are in the in-between set of $a_{s,2}$ and $a_{s+1,2}$. Since p is productive, we have that p covers $\{a_{r-1,2}, a_{2,2}\}$ by Lemma 2.
Case 2: r is odd and $s \neq r-2$. It again follows from from the definition of $L_{r,f}$ and from Lemma 2 that $p \lhd \{a_{r-1,2}, a_{2,2}\}$.
Case 3: r is odd and $s = r-2$. Then $A_2 \lhd \{a_{r-2,2}, a_{r-1,2}\}$. By Lemma 2 and from the definition of $L_{r,f}$, p must also cover $a_{3,2}$; by Lemma 1 p covers $\overline{a_3}$. One more time we restrict our attention to the case in which $A_1 \lhd a_{3,1}$ and $A_1 \lhd a_{s,1}$. Since $s \geq 4$, we can apply the same reasoning to see that p covers $\overline{a_4}$. But since $3 \neq r-2$, we are now in Case 2.

We may conclude that $p \lhd \{a_{r-1,2}, a_{2,2}\}$. By Lemma 1, $a_{2,1}$ and $a_{r-1,1}$ must also be covered by p. The only interesting case is if they are covered

by A_1. But then by Lemma 2 $p \lhd \{a_{j,1} | 2 \leq j \leq r-1\}$, and we are done by Lemma 1. ■

The following lemma presents a property of derivations in G that will be used to "factorize" sentential derivations for sentences in $L_M(G)$. We need to introduce two additional notions. Let p be a production whose left-hand tuple is $(A_1, \ldots, A_t)$, $t \leq f$. Assume the existence of a sentential derivation of the form

$$(S, I^{\langle S \rangle}) \stackrel{*}{\Rightarrow}_G (u_0 A_1 u_1 \cdots u_{t-1} A_t u_t, I^{\langle A_1, \ldots, A_t \rangle}),$$

where $u_i \in (V_{\mathrm{T}}^{(r,f)})^*$. Then $u_0 A_1 u_1 \cdots u_{t-1} A_t u_t$ is called a *p-factorized* sentential form. Let a, b, c be different symbols in $V_{\mathrm{T}}^{(r,f)}$. We say that b is *isolated* in the above sentential form whenever, for strings $x, y, v, z \in (V_{\mathrm{T}}^{(r,f)})^*$, one of the following conditions is realized: (i) $u_0 = xbycv$, (ii) $u_j = xaybvcz$ for some j, $1 \leq j \leq t-1$, or (iii) $u_t = xaybv$. Note that whenever a terminal symbol a is isolated in a p-factorized sentential form, then p cannot generate a.

Lemma 5 *Let $f \geq 2$, $r \geq 6$. Let p be a productive production such that $p \lhd \mathcal{B}^{(r,f)}$ and let $u_0 A_1 u_1 \cdots u_{t-1} A_t u_t$, $t \leq f$, be a p-factorized sentential form. Then for every terminal group $\overline{a} \in \mathcal{B}^{(r,f)}$ there exists a terminal symbol $a \in \overline{a}$ such that a is not found in string $u_0 u_1 \cdots u_t$.*

Proof. For the sake of contradiction, assume that $u_0 u_1 \cdots u_t$ contains instances of every terminal symbol in some $\overline{a_s} \in \mathcal{B}^{(r,f)}$. First of all, we claim that no u_i, $0 \leq i \leq t$, can contain two different terminals from $\overline{a_s}$. From the definition of $L_{r,f}$ it can be seen that for any $a_{s,q}$, $a_{s,q'}$ in $\overline{a_s}$, $q \neq q'$, the in-between set of $a_{s,q}$ and $a_{s,q'}$ contains at least one terminal b from some $\overline{a_{s'}} \in \mathcal{B}^{(r,f)}$, $s' \neq s$. If $a_{s,q}$ and $a_{s,q'}$ are included in u_i, then b will be isolated in the p-factorized sentential form and p could not generate b, contrary to the hypotheses. This proves our claim.

Let $l = \pi^{-1}(r-1)$, i.e., $l = r-2$ if r is even, $l = r-3$ if r is odd. To prove the lemma, we will distinguish three cases.
Case 1: $s \notin \{2, l\}$. If any $a_{s,q} \in \overline{a_s}$ is included in u_0, then $a_{2,1}$ to its left will be isolated and p could not generate $a_{2,1}$, contrary to the hypotheses. Similarly, if any $a_{s,q}$ is included in u_t, then $a_{l,f}$ to its right will be isolated and p could not generate $a_{l,f}$, again a contradiction. We conclude therefore that terminals in $\overline{a_s}$ are all contained within $u_1 \cdots u_{t-1}$. Since $t \leq f$, there

will be some u_i, $1 \leq i \leq t-1$, which contains two different terminals from $\overline{a_s}$, contradicting our claim.
Case 2: $s = 2$. If any $a_{2,q} \in \overline{a_2}$ is contained in u_t, then $a_{l,f}$ to its right will be isolated and p could not generate such a symbol. From our claim and the definition of $L_{r,f}$, it follows that $t = f$ and each $a_{2,q}$ is contained in u_{q-1} for $1 \leq q \leq f$. Consider now A_1. Since p covers $\mathcal{B}^{(r,f)}$ and u_1 contains $a_{2,2}$, A_1 must cover at least $a_{2,1}$ and $a_{3,2}$ (which occur to the left of $a_{2,2}$ in strings of $L_{r,f}$). Since p is productive, by Lemma 2 p covers the in-between set of $a_{2,1}$ and $a_{3,2}$, which includes $a_{1,2}$; therefore p covers $\overline{a_1}$ by Lemma 1. But this is impossible, since u_0 must contain at least one instance of $a_{1,1}$ to the left of $a_{2,1}$, which is therefore isolated.
Case 3: $s = l$. If any $a_{l,q} \in \overline{a_l}$ is contained in u_0, then $a_{2,1}$ to its left will be isolated and p could not generate such a symbol. Again it follows from our claim that $t = f$ and each $a_{l,q}$ is contained in u_q, for $1 \leq q \leq f$. Consider A_2. Since p covers $\mathcal{B}^{(r,f)}$, u_1 contains $a_{l,1}$ and u_2 contains $a_{l,2}$, A_2 must cover at least $a_{r-1,1}$ and $a_{3,2}$ (which occur in between $a_{l,1}$ and $a_{l,2}$ in strings of $L_{r,f}$). Since p is productive, by Lemma 2 p covers the in-between set of $a_{r-1,1}$ and $a_{3,2}$, which includes $a_{r,1}$; therefore p covers $\overline{a_r}$ by Lemma 1. This is impossible, because u_f must contain at least one instance of $a_{r,f}$ to the right of $a_{l,f}$, which is therefore isolated. ■

Let p be a production satisfying the hypotheses of Lemma 5 and let p', p'' be any pair of productions that can simultaneously be used to rewrite the right-hand tuple of p. As a consequence of Lemma 5, we have that whenever p' covers $\mathcal{B}^{(r,f)}$, p'' cannot cover any of the terminal groups in $\mathcal{B}^{(r,f)}$. This observation will be used in the proof of the following theorem, which refers to all previous results. The theorem shows that, for all sentences w in some subset of $L_{r,f}$, any derivation in G of w can be partitioned into two parts. In a sense to be made more precise below, the first part of the derivation cannot generate all terminal symbols in any terminal group in $\mathcal{B}^{(r,f)}$, while the second part of the derivation uses productions that do cover $\mathcal{B}^{(r,f)}$.

Theorem 1 *Let $f \geq 2$, $r \geq 6$. Then we have $L_{r,f} \in \mathcal{L}((r-2)\text{-LUSCG}(f)) - \mathcal{L}((r-3)\text{-LUSCG}(f))$.*

Proof. A grammar in $(r-2)$-LUSCG(f) that derives $L_{r,f}$ has been presented in Example 2. To prove the statement, we show that the assumption of the existence of $G \in (r-3)$-LUSCG(f) such that $L(G) = L_{r,f}$ leads to a contradiction.

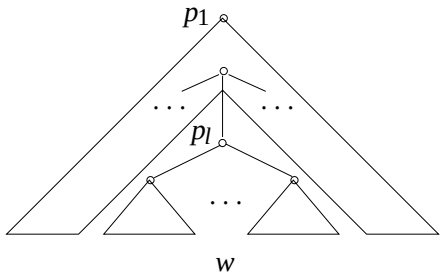


Figure 5: A derivation of w in G, represented as a derivation tree in $\mathrm{der}(G)$. The part of the tree above production p_l cannot generate all terminal symbols of any of the terminal groups in $\mathcal{B}^{(r,f)}$. By the construction of w, these symbols must therefore be covered by the productions that are the daughters of p_l.

Let Δ_G be the maximum number of terminal symbols in the right-hand tuple of a production of G. Let w be a sentence in $L_{r,f}$ such that $\#_a(w) > (r-3) \cdot M_G + \Delta_G$ for every $a \in V_{\mathrm{T}}^{(r,f)}$, and let also ρ be a derivation in G for w. Note that every production in ρ is productive. We uniquely identify a production used in ρ in the following way. Let p_1 be the first production used in ρ, i.e., ρ has the form $(S, I^{\langle S \rangle}) \overset{p_1}{\Rightarrow}_G (\alpha, I') \overset{*}{\Rightarrow}_G (w, I'')$. S is a p_1-factorized sentential form and, by the choice of w, p_1 covers $\mathcal{B}^{(r,f)}$. Let $p_{1,1}, \ldots, p_{1,k_1}$, $1 \leq k_1 \leq r-3$, be the sequence of productions used in ρ to rewrite the right-hand tuple of p_1. As already observed, Lemma 5 entails that at most one production in such a sequence can cover $\mathcal{B}^{(r,f)}$. If such a production exists, we call it p_2. We iterate the step until we arrive at some production p_l, $l \geq 1$, used in ρ such that p_l covers $\mathcal{B}^{(r,f)}$ and none of the productions that are used in ρ to rewrite the right-hand tuple of p_l (if any) covers $\mathcal{B}^{(r,f)}$ (see Figure 5).

Let $u_0 A_1 u_1 \cdots u_{t-1} A_t u_t$ be the p_l-factorized sentential form defined by p_l. Since p_l is productive, we invoke Lemma 5 and conclude that, for every terminal group $\overline{a_s} \in \mathcal{B}^{(r,f)}$, there exists a terminal a_{s,q_s} that is not contained within string $u_0 u_1 \cdots u_t$. Hence, more than $(r-3) \cdot M_G + \Delta_G$ instances of each a_{s,q_s}, $2 \leq s \leq r-1$, are generated under ρ from the non-terminals in the right-hand tuple of p_l. Now let $p_{l,1}, \ldots, p_{l,k_l}$, $1 \leq k_l \leq r-3$, be the sequence of productions used in ρ to rewrite the right-hand tuple of p_l (clearly this sequence cannot be empty). The right-hand tuple of p_l

itself cannot contain more than Δ_G instances of each a_{s,q_s}, and therefore $p_{l,1},\ldots,p_{l,k_l}$ must generate more than $(r-3)\cdot M_G$ instances of each a_{s,q_s}. Since $k_l \leq r-3$, by a counting argument we conclude that for each s, $2 \leq s \leq r-1$, there must be at least one $p_{l,i}$, $1 \leq i \leq k_l$, such that $p_{l,i}$ generates more than M_G instances of a_{s,q_s} and hence covers a_{s,q_s}. By Lemma 1, $p_{l,i}$ covers $\overline{a_s}$. Again by a counting argument, we derive that at least one production $p_{l,i}$, $1 \leq i \leq k_l$, covers two terminal groups in $\mathcal{B}^{(r,f)}$. Since $p_{l,i}$ is productive, it covers $\mathcal{B}^{(r,f)}$ by Lemma 4. This contradicts the choice of production p_l: we conclude that there can be no derivation in G for w, that is, grammar G does not exist. ■

We now turn to subclasses 2-LUSCG(f) and 3-LUSCG(f), $f \geq 2$. We first show that for $f = 2$, they collapse.

Theorem 2 $\mathcal{L}(\text{2-LUSCG}(2)) = \mathcal{L}(\text{3-LUSCG}(2))$.

Proof. We show how to convert a grammar $G \in \text{3-LUSCG}(2) - \text{2-LUSCG}(2)$ into $G' \in \text{2-LUSCG}(2)$ such that $L(G) = L(G')$. Let p be a production of G of order three. Assume p is of the form $(A_1, A_2) \rightarrow (\alpha_1, \alpha_2)$ with $\alpha_1 = u_0 B_1 u_1 \cdots u_{l-1} B_l u_l$ and $\alpha_2 = v_0 C_1 v_1 \cdots v_{r-1} C_r v_r$, where $B_i, C_i \in V_{\text{N}}$, $u_i, v_i \in V_{\text{T}}^*$, and where l, r are nonnegative integers with $3 \leq l + r \leq 6$. Let $\Pi_p = \{p_1, p_2, p_3\}$ be any multiset of productions of G that rewrites the right-hand tuple of p. We distinguish three cases.
Case 1: $l, r > 1$. By a counting argument, there must be $p_h \in \Pi_p$ such that $\varphi(p_h) = 2$ and p_h rewrites two nonterminals among B_1, B_l, C_1 and C_r. Assume p_h rewrites B_l and C_1. For new nonterminal symbols $[p, p_h, 1]$ and $[p, p_h, 2]$, construct productions

$$\begin{aligned} (A_1, A_2) &\rightarrow ([p, p_h, 1] B_l u_l,\ \ v_0 C_1 [p, p_h, 2]), \\ ([p, p_h, 1], [p, p_h, 2]) &\rightarrow (u_0 B_1 \cdots B_{l-1} u_{l-1},\ \ v_1 C_2 \cdots C_r v_r) \end{aligned} \tag{4}$$

to be used by G'. By assumption, productions in (4) have order not greater than two. The remaining cases for p_h are handled in a similar way.
Case 2: $l = 1$ or $r = 1$. Assume $l = 1$. Choose production $p_h \in \Pi_p$ such that p_h rewrites B_1. If $\varphi(p_h) = 2$, p_h also rewrites nonterminal C_q for some $1 \leq q \leq r$. If $q \notin \{1, r\}$, then for new nonterminal symbols $[p, p_h, 1]$ and $[p, p_h, 2]$ construct productions

$$\begin{aligned} (A_1, A_2) &\rightarrow (u_0 B_1 u_1,\ \ [p, p_h, 1] C_q [p, p_h, 2]), \\ ([p, p_h, 1], [p, p_h, 2]) &\rightarrow (v_0 C_1 \cdots C_{q-1} v_{q-1},\ \ v_q C_{q+1} \cdots C_r v_r). \end{aligned} \tag{5}$$

Productions in (5) have order not greater than two. If $q = 1$, $q = r$ or $\varphi(p_h) = 1$, we have subcases that can be handled with just one new nonterminal.
Case 3: $l = 0$ or $r = 0$. Assume $l = 0$. Choose production $p_h \in \Pi_p$ such that p_h rewrites C_1. If $\varphi(p_h) = 2$, p_h also rewrites nonterminal C_q for some $2 \leq q \leq r$. If $q \notin \{2, r\}$, then for new nonterminal symbols $[p, p_h, 1]$ and $[p, p_h, 2]$ construct productions

$$\begin{aligned} (A_1, A_2) &\rightarrow (u_0,\ v_0 C_1 [p, p_h, 1] C_q [p, p_h, 2]), \\ ([p, p_h, 1], [p, p_h, 2]) &\rightarrow (v_1 C_2 \cdots C_{q-1} v_{q-1},\ v_q C_{q+1} \cdots C_r v_r). \end{aligned} \quad (6)$$

Again, productions in (6) have order not greater than two. If $q = 2$, $q = r$ or $\varphi(p_h) = 1$, we have two subcases that can be handled with just one new nonterminal. This exhaust all cases in which $\varphi(p) = 2$.
Finally, if p is of the form $(A) \rightarrow (\alpha)$, we can proceed as in Case 3 above. ■

Next we will show that for any integer $f \geq 3$, the class 2-LUSCG(f) is properly included in 3-LUSCG(f). The family of languages studied at the beginning of this section cannot be used in order to prove this separation result, and we have to define new languages to which we will restrict our attention in what follows.

Definition 7 *Let f be an integer, $f \geq 3$, and let $V_{\mathrm{T}}^{(f)} = \{a_{1,h}, a_{2,h}, a_{3,h}, a_{4,h}, a_{5,h} \mid 1 \leq h \leq f\}$. Language Q_f is specified as follows:*

$$\begin{aligned} Q_f = \{ & w_1 w_2 \cdots w_f \mid w_1 = a_{1,1}^{i_1} a_{2,1}^{i_2} a_{3,1}^{i_3} a_{4,1}^{i_4} a_{5,1}^{i_5},\ w_2 = a_{1,2}^{i_1} a_{3,2}^{i_3} a_{2,2}^{i_2} a_{4,2}^{i_4} a_{5,2}^{i_5}, \\ & w_h = a_{1,h}^{i_1} a_{2,h}^{i_2} a_{4,h}^{i_4} a_{3,h}^{i_3} a_{5,h}^{i_5},\ 3 \leq h \leq f, i_j \geq 1,\ 1 \leq j \leq 5 \}. \end{aligned}$$

As in the case of languages $L_{r,f}$, we will call *segment* each substring w_h, $1 \leq h \leq f$, in the definition of a string in Q_f. We will also use the terminal group notation $\overline{a_s} = \{a_{s,1}, a_{s,2} \ldots, a_{s,f}\}$, $1 \leq s \leq 5$. Finally, the set of all terminal groups for Q_f will be denoted $\mathcal{V}^{(f)}$ and the set $\{\overline{a_2}, \overline{a_3}, \overline{a_4}\}$ will be denoted $\mathcal{B}^{(f)}$. We now study some properties that are common to all grammars in LUSCG that derive languages Q_f. (Henceforth, we will always assume $f \geq 3$.) In what follows there is a strong similarity with the properties of languages $L_{r,f}$ that have been investigated so far; for this reason, sometimes proofs will be omitted; in the remaining cases, our arguments will be simpler than those used for languages $L_{r,f}$, due to the fact that languages Q_f depend upon only one parameter.

Assume that $G \in$ LUSCG is a grammar deriving some language Q_f. We start with three properties of G that correspond to Lemmas 1, 2 and 3. Let a and b be two symbols in $V_{\mathrm{T}}^{(f)}$. For any string w in Q_f, the set of all terminal symbols occurring between a and b in w (including a and b) is always the same. Again such a set will be called the *in-between set* of a and b.

Lemma 6 *For every production p of G, the following statements hold:*

(i) *if p covers some $a_{s,q}$ then p covers $\overline{a_s}$;*

(ii) *if p is productive and if a nonterminal symbol A in the left-hand tuple of p covers (by means of p) set $\{a, b\}$, then p covers the in-between set of a and b;*

(iii) *if p is productive and if a nonterminal symbol A in the left-hand tuple of p covers (by means of p) set $\{a_{s,q}, a_{s,q'}\}$, then p covers all of the terminals in $\mathcal{V}^{(f)}$.*

Proof. Statements (i) and (ii) can be proved using the same arguments found in the proofs of Lemmas 1 and 2. We prove here statement (iii). Assume that $q < q'$ and that $(A_1, \ldots, A_t)$, $1 \geq t \leq f$, is the left-hand tuple of p. Observe that symbols $a_{5,q}$ and $a_{1,q+1}$ are both included in the in-between set of $a_{s,q}$ and $a_{s,q'}$. Then by statement (ii) of the lemma p must cover these symbols and by statement (i) p must cover $\overline{a_1}$ and $\overline{a_5}$. If one nonterminal B from the left-hand tuple of p covers more than one member of $\overline{a_1}$, say $a_{1,u}$ and $a_{1,u'}$, by statement (ii) we have $p \lhd a_{j,u}$ for $1 \leq j \leq 5$ and hence, by statement (i), $p \lhd \mathcal{V}^{(f)}$. Suppose instead that each nonterminal from the left-hand tuple of p covers exactly one member of $\overline{a_1}$, which means $t = f$. Let B be the nonterminal that covers $a_{1,1}$. If B covers any member of $\overline{a_5}$, then by statement (ii) p covers $a_{j,1}$ for $1 \leq j \leq 5$, and by statement (i) p covers $\mathcal{V}^{(f)}$. If B does not cover any member of $\overline{a_5}$, there must be a nonterminal from the left-hand tuple of p which covers two members of $\overline{a_5}$. Again we are done by statements (ii) and (i). ■

We now derive a basic property of productive productions in G. What follows is the analogue of Lemma 4 for languages Q_f.

Lemma 7 *If a productive production p of G covers more than one terminal group in $\mathcal{B}^{(f)}$, then p covers $\mathcal{B}^{(f)}$.*

Proof. Let p cover groups $\overline{a_s}$ and $\overline{a_{s'}}$ in $\mathcal{B}^{(f)}$, $s < s'$; assume also that $(A_1, \ldots, A_t)$, $1 \leq t \leq 1$, is the left-hand tuple of p. If there exists i, $1 \leq i \leq t$, such that A_i covers two different terminal symbols in $\overline{a_s}$ (with respect to p) or A_i covers two different terminal symbols in $\overline{a_{s'}}$, we are done by statement (iii) in Lemma 6. Suppose instead that $t = f$ and, for every $1 \leq i \leq f$, A_i covers exactly one symbol in terminal group $\overline{a_s}$ and exactly one symbol in terminal group $\overline{a_{s'}}$. Without loss of generality we can assume that A_i covers $a_{s,i}$ for each i. From the definition of Q_f it follows that, for each i, A_i also covers $a_{s',i}$. There is a finite number of cases for the pair s, s': again from the definition of Q_f we see that in all cases a terminal symbol $a_{s'',i}$ is included in the in-between set of $a_{s,i}$ and $a_{s',i}$ for some i, where $\overline{a_{s''}} \in \mathcal{B}^{(f)}$ and $s'' \notin \{s, s'\}$. Since p is productive, by Lemma 6 p must cover $a_{s'',q}$ and therefore $\overline{a_{s''}}$. This concludes the proof. ■

The notion of p-factorized sentential form for a production p and the associated notion of isolated symbol have been introduced in the discussion preceding Lemma 5. These notions will also be used in the following statement, which represents for languages Q_f the analogue of Lemma 5.

Lemma 8 *Let p be a productive production of G such that $p \lhd \mathcal{B}^{(f)}$ and let $u_0 A_1 u_1 \cdots u_{t-1} A_t u_t$, $t \leq f$, be a p-factorized sentential form. Then for every terminal group $\overline{a} \in \mathcal{B}^{(f)}$ there exists a terminal symbol $a \in \overline{a}$ such that a is not found in string $u_0 u_1 \cdots u_t$.*

Proof. We assume that $u_0 u_1 \cdots u_t$ contains instances of every terminal symbol in some $\overline{a_s} \in \mathcal{B}^{(f)}$ and derive a contradiction. First, we claim that no u_i, $0 \leq i \leq t$, can contain two different terminals from $\overline{a_s}$. Assume the contrary. From the definition of Q_f it can be seen that at least one terminal from some $\overline{a_{s'}} \in \mathcal{B}^{(r,f)}$, $s' \neq s$, will be isolated in the p-factorized sentential form. Then p could not generate it, contrary to the hypotheses. To prove the lemma, we then proceed by distinguishing three cases.
Case 1: $s = 4$. If any $a_{4,q} \in \overline{a_4}$ is included in u_0, then $a_{2,1}$ to its left will be isolated and p could not generate $a_{2,1}$, contrary to the hypotheses. A similar argument applies if any $a_{4,q}$ is included in u_t. Since $t \leq f$, we conclude that there is some u_i, $1 \leq i \leq t-1$, which contains at least two different terminals from $\overline{a_4}$. But this contradicts the above claim.
Case 2: $s = 2$. If any $a_{2,q} \in \overline{a_2}$ is contained in u_t, then $a_{3,f}$ to its right will be isolated and p could not generate such a symbol. If $t < f$ we establish a contradiction using again the claim above. Assume therefore $t = f$ and

each $a_{2,q}$ is contained in u_{q-1} for $1 \leq q \leq f$. Since p covers $\mathcal{B}^{(f)}$, u_0 contains $a_{2,1}$ and u_1 contains $a_{2,2}$, the only nonterminal in the left-hand tuple of p that can cover $a_{3,1}$ and $a_{3,2}$ (which occur between symbols $a_{2,1}$ and $a_{2,2}$ in strings in Q_f) is A_1. Since p is productive, by statement (iii) of Lemma 6 p covers $\overline{a_1} \in \mathcal{V}^{(f)}$. But this is impossible, since u_0 must contain at least one instance of $a_{1,1}$ to the left of $a_{2,1}$, which is therefore isolated.
Case 3: $s = 3$. If any $a_{3,q} \in \overline{a_3}$ is contained within u_0, then $a_{2,1}$ to its left will be isolated and p could not generate such a symbol. Again we deal with the case $t = f$ and $a_{3,q}$ in u_q for $1 \leq q \leq f$. With an argument similar to Case 2, we can argue that p covers $\overline{a_5} \in \mathcal{V}^{(f)}$. Again this is not possible, because u_f must contain at least one instance of $a_{5,f}$ to the right of $a_{3,f}$, which is therefore isolated. ■

The technique used in the proof of Theorem 1 along with the above lemmas can be used to show the following result. The proof is omitted because of its strong similarity with the one of Theorem 1.

Theorem 3 *Let f be an integer, $f \geq 3$. Then we have $Q_f \in \mathcal{L}(2\text{-LUSCG}(f)) - \mathcal{L}(3\text{-LUSCG}(f))$.* ■

To conclude this section and to complete our picture of the rank hierarchy for fixed values of the fan-out parameter, we give a last result that compares the subclasses of LUSCG of ranks one and two. The proof of the result, however, must be deferred to Section 5, where we will use results of Section 4 along with an equivalence result that allows us to transfer to LUSCG some facts that are already known for the class of parallel rewriting systems.

Theorem 4 *Let $f \geq 1$. Then $\mathcal{L}(1\text{-LUSCG}(f))$ is properly included in $\mathcal{L}(2\text{-LUSCG}(f))$.* ■

Section 5 will complete our investigation of the interaction between the fan-out and rank complexity measures by transferring the rank hierarchy results of this section to parallel rewriting systems and combining them with a fan-out hierarchy result that is well known for the latter class.

4 Closure properties

This section investigates some language-theoretic properties of classes r-LUSCG(f), $r, f \geq 1$. We will use these results in Section 5.

A family of languages is called an abstract family of languages, for short AFL, if it is closed under union, concatenation, ε-free Kleene closure, ε-free homomorphism, inverse homomorphism and intersection with regular languages. A full AFL is an AFL which is also closed under arbitrary homomorphism.

Theorem 5 *For integers $r \geq 2$ and $f \geq 1$, $\mathcal{L}(r\text{-LUSCG}(f))$ is a substitution-closed full* AFL.

Proof. Since for $r \geq 2$ and $f \geq 1$, r-LUSCG(f) contains all regular languages, it is sufficient to show closure under substitution and under intersection with regular languages [Salomaa, 1973, p.126].

To show closure under substitution, consider a grammar G in r-LUSCG(f), $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, P, S)$. Let δ_G be the length of the longest sequence of consecutive terminal symbols introduced by a rule in P. We will construct a new grammar $G' = (V'_{\mathrm{N}}, V_{\mathrm{T}}, P', [S])$ as follows. Let $V'_{\mathrm{N}} = \{[uAv], [uAv, p] \mid A \in V_{\mathrm{N}}, u, v \in V_{\mathrm{T}}^*, |u|, |v| \leq \delta_G \text{ and } p \in P\}$. Let also p be a production in P of the form $(A_1, \ldots, A_t) \to (\alpha_1, \ldots, \alpha_t)$, where $1 \leq t \leq f$ and for $1 \leq k \leq t$ we have $\alpha_k = u_{k,0} B_{k,1} u_{k,1} \cdots u_{k,l_k-1} B_{k,l_k} u_{k,l_k}$, $l_k \geq 0$, $B_{k,i} \in V_{\mathrm{N}}$ for $1 \leq i \leq l_k$ and $u_{k,j} \in V_{\mathrm{T}}^*$ for $0 \leq j \leq l_k$. For every tuple $\tau = (u_1, v_1, \ldots, u_t, v_t)$ such that $u_i, v_i \in V_{\mathrm{T}}^*$ and $|u_i|, |v_i| \leq \delta_G$, $1 \leq i \leq t$, we add to P' the production

$$p_\tau : ([u_1 A_1 v_1], \ldots, [u_t A_t v_t]) \to \\ ([u_1 A_1 v_1, p], \ldots, [u_t A_t v_t, p]).$$

Furthermore, for every τ as above and for every $a \in V_{\mathrm{T}}$, $X \in \{a, S\}$ and $1 \leq h \leq t$, we add to P' the production

$$p_{\tau,X} : ([A_1, p], \ldots, [A_{h-1}, p], [a u_h A_h v_h, p], \ldots, [u_t A_t v_t, p]) \to \\ ([A_1, p], \ldots, [A_{h-1}, p], X[u_h A_h v_h, p], \ldots, [u_t A_t v_t, p])$$

and the production

$$p'_{\tau,X} : ([A_1, p], \ldots, [A_{h-1}, p], [u_h A_h v_h a, p], \ldots, [u_t A_t v_t, p]) \to \\ ([A_1, p], \ldots, [A_{h-1}, p], [u_h A_h v_h, p]X, \ldots, [u_t A_t v_t, p]).$$

Finally, we add to P' the production

$$p' : ([A_1, p], \ldots, [A_t, p]) \quad \to \quad (\alpha'_1, \ldots, \alpha'_t),$$

where $\alpha'_k = [u_{k,0}A_{k,1}][u_{k,1}A_{k,2}]\ldots[u_{l_k-1}A_{k,l_t}u_{k,l_t}]$, $1 \leq k \leq t$. The process is iterated for every p in P.

It is straightforward to show that G' generates the substitution closure of G; we omit the details. We have $\rho(p') = \rho(p)$ and $\rho(p_\tau) = \rho(p_{\tau,X}) = \rho(p'_{\tau,X}) \leq 2$ for every p in P. Thus, $\rho(G') = \rho(G)$. Since all productions in P' derived from p in P preserve the fan-out of p, we have $G' \in r\text{-LUSCG}(f)$.

As far as intersection with regular languages is concerned, we anticipate here some of the contents of the next section (Theorem 6), where an equivalence result is presented between classes r-LUSCG(f) and classes r-MCFG(f) studied in [Seki *et al.*, 1991]. In [Seki *et al.*, 1991, Theorem 3.9] it is shown that, for every $f \geq 1$, $\cup_{r\geq 1} r\text{-MCFG}(f)$ is closed under intersection with regular languages; their proof preserves parameter r. Hence our result follows from Theorem 6. ■

We obtain the following two corollaries, the first of which was proven (more simply) in [Seki *et al.*, 1991, Theorem 3.9].

Corollary 1 *For $f \geq 1$, $\mathcal{L}$(LUSCG(f)) is a substitution-closed full* AFL. ■

Corollary 2 *For $r \geq 2$, $\mathcal{L}(r\text{-LUSCG})$ is a substitution-closed full* AFL. ■

5 A two-dimensional hierarchy

In this section we provide an overview over some classes of finite copying parallel rewriting systems that have been defined in the literature. We start by proving a generative equivalence relation between these formalisms and the class LUSCG. The importance of such a result is that it provides an original characterization of finite copying parallel rewriting systems in terms of the locality restriction that was introduced in Section 2. At the same time, the equivalence result maps the fan-out and rank parameters defined for LUSCG into synchronous parallelism and independent parallelism respectively, as defined for the parallel rewriting systems we consider here. In this way we can transfer the results presented so far and show how independent parallelism induces an infinite non-collapsing hierarchy in parallel rewriting systems with degree of synchronous parallelism bounded by a constant greater than one. Finally, we combine this hierarchy result with already known properties of parallel rewriting systems and show the existence of a

two-dimensional infinite hierarchy induced by the synchronous and the independent parallelism parameters. This provides a new picture of the class of finite copying parallel rewriting systems. The hierarchy result, combined with the results of Section 4, also provides an answer to a question that was left open in the literature. In what follows, we will use the term *rank* of a context-free grammar to refer to the greatest number of nonterminal symbols that can be found in the right-hand side of the productions of the grammar.

We start by relating class LUSCG to a class of rewriting systems known as multiple context-free grammars (MCFG) introduced in [Kasami *et al.*, 1987; Seki *et al.*, 1991]. For notational convenience, we present MCFG through a notational variant of this class that in [Weir, 1992] is called string-based linear context-free rewriting system. This variant requires the "information-lossless" condition (see [Seki *et al.*, 1991]) while MCFG does not. However, [Seki *et al.*, 1991] show that this does not affect the generative power of the class (their Lemma 2.2).[1] We discuss the relationship between LUSCG and MCFG in some detail, since existing results will then allow us to relate LUSCG to other known formalisms as well.

Let V_{T} be an alphabet of terminal symbols; in the following we will be interested in functions mapping tuples of strings in V_{T}^* into tuples of strings in V_{T}^*. For integers r and f, $r \geq 0$ and $f \geq 1$, we say that g is an r-ary function if there exist integers $f_i \geq 1$, $1 \leq i \leq r$, such that g is defined on $(V_{\mathrm{T}}^*)^{f_1} \times (V_{\mathrm{T}}^*)^{f_2} \times \cdots \times (V_{\mathrm{T}}^*)^{f_r}$; we say that g has fan-out f if the range of g is a subset of $(V_{\mathrm{T}}^*)^f$. Let y_h, x_{ij}, $1 \leq h \leq f$, $1 \leq i \leq r$ and $1 \leq j \leq f_r$, be string-valued variables. A function g as above is said to be *linear regular* if it is defined by an equation of the form

$$g(\langle x_{1,1}, \ldots, x_{1,k_1} \rangle, \ldots, \langle x_{r,1}, \ldots, x_{r,k_r} \rangle) \quad = \quad \langle y_1, \ldots, y_f \rangle, \tag{7}$$

where $\langle y_1, \ldots, y_f \rangle$ represents some grouping into f sequences of all and only the variables appearing in the left-hand side of (7) (without repetitions) along with some additional terminal symbols (with possible repetitions). The following definition is based on [Weir, 1992, p.137] and [Seki *et al.*, 1991, p.196], and can easily seen to be a notational variant of either.

[1] String-based linear context-free rewriting system is a member of the family of linear context-free rewriting system (LCFRS) introduced in [Vijay-Shanker *et al.*, 1987; Weir, 1988] independently of MCFG. This family groups together a large class of rewriting systems that operate on different types of objects, such as strings, tuples of strings, trees, graphs, and so on. The result of rewriting is then associated with terminal strings by "yield functions", in order to generate string languages.

Definition 8 *A* multiple context-free grammar *(*MCFG*) is a quadruple* $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, P, S)$ *where* V_{N}, V_{T} *and* S *are defined as for an unordered scattered context grammar, every symbol* $A \in V_{\mathrm{N}}$ *is associated with an integer* $\varphi(A) \geq 1$ *and* P *is a finite set of productions of the form* $p : A \to g(B_1, B_2, \ldots, B_{\rho(p)})$, *where* $\rho(p) \geq 0$, $A, B_i \in V_{\mathrm{N}}$, $1 \leq i \leq \rho(p)$ *and where* g *is a linear regular function having arity* $\rho(p)$ *and fan-out* $\varphi(A)$, *defined on* $(V_{\mathrm{T}}^*)^{\varphi(B_1)} \times \cdots \times (V_{\mathrm{T}}^*)^{\varphi(B_{\rho(p)})}$.

For every $A \in V_{\mathrm{N}}$, *we write* $A \Rightarrow_G \langle y_1, \ldots, y_{\varphi(A)} \rangle$, $\langle y_1, \ldots, y_{\varphi(A)} \rangle \in (V_{\mathrm{T}}^*)^{\varphi(A)}$, *if one of the following conditions is met:*

(i) $A \to g() \in P$ *and* $g() = \langle y_1, \ldots, y_{\varphi(A)} \rangle$;

(ii) $A \to g(B_1, \ldots, B_{\rho(p)}) \in P$, $B_i \Rightarrow_G t_i$ *for every* $1 \leq i \leq \rho(p)$, *where* $t_i \in (V_{\mathrm{T}}^*)^{\varphi(B_i)}$, *and* $g(t_1, \ldots, t_{\rho(p)}) = \langle y_1, \ldots, y_{\varphi(A)} \rangle$.

We emphasize that in MCFG the rewrite relation is only defined for tuples of terminal strings. For $A \in V_{\mathrm{N}}$, we call $\varphi(A)$ the *fan-out* of A; for $p \in P$, we call $\rho(p)$ the *rank* of p and we write $\varphi(p) = \varphi(A)$ whenever A is the left-hand side symbol of p. For $G \in$ MCFG, we define $\varphi(G) = \max_{A \in V_{\mathrm{N}}} \varphi(A)$ and $\rho(G) = \max_{p \in P} \rho(p)$. For $r \geq 0$ and $f \geq 1$, the class of all linear context-free rewriting systems with rank bounded by r and fan-out bounded by f is denoted r-MCFG(f).[2] The language derived by G is the set of tuples $L(G) = \{\langle y_1, \ldots, y_{\varphi(S)} \rangle \mid S \Rightarrow_G \langle y_1, \ldots, y_{\varphi(S)} \rangle\}$. Without any loss of generality, in what follows we assume that there are no useless nonterminals in G and that $\varphi(S) = 1$.

Example 3 Let L be the language considered in Example 1. A grammar $G'_L \in$ 2-MCFG(2) that generates L is defined in Figure 6. ■

The following theorem establishes a strong (rank- and fan-out-preserving) equivalence relation between LUSCG and MCFG. The proof is conceptually straightforward but notationally complex; we defer it to Appendix A.

Theorem 6 *Let* r, f *be integers such that* $r, f \geq 1$. *Then we have* $\mathcal{L}(r\text{-MCFG}(f)) = \mathcal{L}(r\text{-LUSCG}(f))$. ■

We can immediately obtain the following rank hierarchy result for MCFG.

[2] [Seki *et al.*, 1991] use the notation f-MCFG to refer to MCFG of fan-out f, while we use r-MCFG to refer to MCFG of rank r.

$$
\begin{array}{rcl}
G'_L & = & (V_{\mathrm{N}}, V_{\mathrm{T}}, S, P); \\
V_{\mathrm{N}} & = & \{S, A, B\} \\
V_{\mathrm{T}} & = & \{[,]\} \\
P & = & \{\mathrm{p}_1\colon \quad S \to g(A), \\
 & & \ \mathrm{p}_2\colon \quad A \to f_1(), \\
 & & \ \mathrm{p}_3\colon \quad A \to f_2(A), \\
 & & \ \mathrm{p}_4\colon \quad A \to f_3(A, A)\ \} \\
g(\langle x_{11}, x_{12}\rangle) & = & \langle x_{11}x_{12}\rangle, \\
f_1() & = & \langle [\,], [\,]\rangle, \\
f_2(\langle x_{11}, x_{12}\rangle) & = & \langle [x_{11}], [x_{12}]\rangle, \\
f_3(\langle x_{11}, x_{12}\rangle, \langle x_{21}, x_{22}\rangle) & = & \langle x_{11}x_{21}, x_{12}x_{22}\rangle.
\end{array}
$$

Figure 6: A multiple context-free grammar for language $L = \{ww | w \in D_1\}$.

Theorem 7 *For each $f \geq 2$, the rank parameter induces a non-collapsing hierarchy in class* MCFG(f).

Proof. The statement directly follows from our main result and from Theorem 6. ■

Next we switch to other finite copying parallel rewriting systems that have been defined in the literature, and use Theorem 6 to transfer our main result to these formalisms. Deterministic tree-walking transducers (DTWT) were introduced by [Aho and Ullman, 1971] (called TAT there). A DTWT transducer is an automaton with a finite state control, that visits in checking mode an input tree generated by a context-free grammar and outputs a translation string. Since this (sequential) device can visit a given subtree more than once, the output tree will contain separated substrings that are "homomorphic" to (a string representation of) that structure. Two complexity measures can be defined for the class DTWT, usually called the *crossing number* and the *rank*. The crossing number of a DTWT represents the maximum number of times the automaton crosses (enters and exits) any subtree in the input tree language; because of the determinism, this number is always finite (see [Aho and Ullman, 1971]). The rank of a DTWT is the rank of the context-free grammar that generates the input language and is finite by definition. For $f \geq 1$ and $r \geq 0$, let us denote by r-DTWT(f) the subclass of all DTWT with crossing number bounded by f and rank bounded by r. If we regard DTWT as generative devices controlled by some tree language, we have the following result.

Theorem 8 *For each $f \geq 2$, the rank parameter induces a non-collapsing hierarchy in class* DTWT(f).

Proof. In [Weir, 1992] it is shown that, for every $f \geq 1$, MCFG(f) has the same generative power as DTWT(f). The proof preserves the rank parameter. The claim then immediately follows from our main result and Theorem 6. ■

Note that in [Aho and Ullman, 1971, p.473] the authors mention the existence of an analogue of the Chomsky normal form for the class DTWT, that is the language produced by a DTWT of any rank can also be obtained by some DTWT of rank two. We remark that this does not contradict Theorem 8, since the conversion into the normal form increases the crossing number. In Theorem 11 below we give a formal proof of their statement, by showing that if the rank is greater than two, it is always possible decrease the rank at the expense of increasing the crossing number.

Top-down tree-to-string transducers (yT) have been introduced in [Engelfriet *et al.*, 1980] as a model of the generalized syntax-directed translation (GSDT) of [Aho and Ullman, 1971] and, in case the degree of independent parallelism is bounded by one, as a model of the controlled ET0L systems of [Rozenberg, 1973]. These parallel rewriting devices take a tree as input, and convert it through a series of rewrite steps into a string. Each rewrite step consumes the root node of a tree in the sentential form, and rearranges the subtrees that are immediately dominated by this node, interleaving them with terminal strings; these subtrees may also be copied. Rewriting is controlled by states which are explicitly represented in the sentential form.

In what follows we regard yT as a class of generative devices controlled by the family of tree languages that can be generated by context-free grammars. With this assumption, two parameters can be defined for these systems. If in a derivation the number of copies of a subtree of the input that a tree-to-string transducer can generate is finite, we say that the transducer has *finite-copying* degree. Furthermore, the *rank* of a transducer is the rank of the context-free grammar that generates the controlling tree language. We denote as r-$yT_{\mathrm{fc}(f)}$ the class of all devices in yT with finite copying degree bounded by f and rank bounded by r, $f, r \geq 1$. The following result can now be easily established.

Theorem 9 *For each $f \geq 2$, the rank parameter induces a non-collapsing hierarchy in class $yT_{\mathrm{fc}(f)}$.*

Proof. In [Engelfriet *et al.*, 1980, Theorem 4.9] it is shown that, for each $f \geq 1$, $\mathcal{L}(yT_{\mathrm{fc}(f)}) = \mathcal{L}(\mathrm{DTWT}(f))$ (the result is achieved using a model called deterministic checking tree transducer). The proof preserves the rank parameter for both classes. Hence the statement follows from Theorem 8. ■

The class of ET0L systems of finite index ($\mathrm{ET0L_{FIN}}$) was introduced by [Rozenberg and Vermeir, 1978; Latteux, 1979]. In [Engelfriet *et al.*, 1980, Theorem 3.2.2] it is shown that, for each $f \geq 1$, the family of languages generated by $\mathrm{ET0L_{FIN}}(f)$ and 1-$yT_{\mathrm{fc}(f)}$ are the same. This gives us the following corollary.

Corollary 3 *For every integer* $f \geq 1$, $1\text{-LUSCG}(f) = \mathrm{ET0L_{FIN}}(f)$.

Using this result, we can now supply the missing proof for Theorem 4 in Section 3, whose statement is repeated here.

Theorem 4 *Let* $f \geq 1$. *Then* $\mathcal{L}(1\text{-LUSCG}(f))$ *is properly included in* $\mathcal{L}(2\text{-LUSCG}(f))$.

Proof. Inclusion holds trivially. We have seen that, for every $f \geq 1$, $\mathcal{L}(2\text{-LUSCG}(f))$ is a substitution-closed AFL (Theorem 5), and thus closed under concatenation. In [Latteux, 1979] it is shown that $\mathcal{L}(\mathrm{ET0L_{FIN}}(f))$ is not closed under concatenation. Properness of the inclusion follows then from Corollary 3. ■

An alternative proof of the above result can be obtained using the well known fact that there exists a context-free language that is not contained in any of the subclasses $\mathcal{L}(\mathrm{ET0L_{FIN}}(f))$, $f \geq 1$ (see [Engelfriet *et al.*, 1980]). But we have already observed that the subclass 2-LUSCG(1) generates all and only the context-free languages.

We can also answer an open question raised in [Engelfriet *et al.*, 1980, p.189], about whether family $\mathcal{L}(yT_{\mathrm{fc}(f)})$ is full principal for each $f \geq 2$.[3]

Corollary 4 *For each* $f \geq 2$, $\mathcal{L}(yT_{\mathrm{fc}(f)})$ *is not full principal.*

Proof. This follows immediately from the the fact that, for every $r \geq 2$ and for every $f \geq 2$, $\mathcal{L}(r\text{-LUSCG}(f))$ is a full AFL (Theorem 5) and from our rank hierarchy result. ■

[3] We are grateful to Joost Engelfriet for drawing our attention to the relevance of our result to this issue.

Context-free hypergraph grammars (CFHG) are rewriting systems that derive sets of edge-labeled hypergraphs; these systems were introduced as a generalization of edge rewriting graph grammars (see for instance [Bauderon and Courcelle, 1987]). In a CFHG, each production specifies some replacement of a labeled hyperedge with a hypergraph, along with particular conditions that allow the replacing hypergraph to be embedded within the host hypergraph. In this way a derivation proceeds sequentially, by replacing hyperedges in a sentential hypergraph; but due to the many tentacles associated with each hyperedge, the derivation can mimic some sort of parallelism. In CFHG, edge rewriting is performed in a context-free fashion, so that the locality restriction is observed. It turns out that each derivation can be associated with an underlying tree structure that can be generated by a context-free grammar (see [Engelfriet and Heyker, 1991]). Two independent parameters can be identified for a CFHG. The first one is the maximum *number of tentacles* associated with a hyperedge in the grammar. The second parameter is the *rank* of the underlying context-free grammar associated with the CFHG. For integers $r, f \geq 1$, we denote by r-CFHG(f) the subclass of all context-free hypergraph grammars whith order bounded by r and maximum number of tentacles bounded by f.

If we restrict attention to CFHG generating string languages, that is chain-like hypergraphs, we find the same generative power as the class DTWT, as shown in [Engelfriet and Heyker, 1991], thus relating CFHG to parallel rewriting systems. We can use their result to transfer our rank hierarchy to CFHG.

Theorem 10 *For each $f \geq 1$, the rank parameter induces a non-collapsing hierarchy in class* CFHG(f).

Proof. In [Engelfriet and Heyker, 1991] it is shown that, for each $f \geq 1$, classes DTWT(f) and (string language generating) CFHG($2f-1$) $\cup$ CFHG($2f$) generate the same family of languages. The proof fails to preserve the rank parameter only in their Lemma 5.3 (p.349). However, in the proof of Theorem 6.5 (p.356) the authors provide an alternative proof of Lemma 5.3 which is in fact rank-preserving. The result then directly follows from our main result and Theorem 8. ■

To conclude the present section, we combine our rank hierarchy result with well known facts about parallel rewriting systems, in order to investigate how synchronous parallelism and independent parallelism interact.

In [Engelfriet *et al.*, 1980] it is observed that each subclass r-$yT_{\mathrm{fc}(1)}$, $r \geq 2$, generates all and only the context-free languages, while subclass 1-$yT_{\mathrm{fc}(1)}$ generates all and only the linear context-free languages. It is well known that the family of linear context-free languages is strictly included in the family of context-free languages (see for instance [Hopcroft and Ullman, 1979]). Again in [Engelfriet *et al.*, 1980] it is observed that, for each $f \geq 1$, there exists a language generated by subclass 1-$yT_{\mathrm{fc}(f+1)}$ that cannot be generated by the class $yT_{\mathrm{fc}(()}f)$. We are then led to the conclusion that the two complexity measures investigated in this work induce a two-dimensional hierarchy for finite copying parallel rewriting systems, that does not collapse. Such a hierarchy is schematicaly represented in Figure 7 by means of an array.

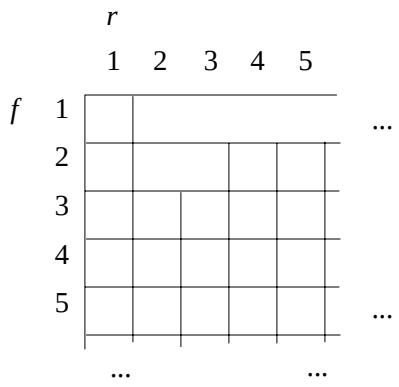


Figure 7: A schematic representation of the two dimensional hierarchy of languages generated by LUSCG and by the finite copying rewriting systems discussed in the present section. The rank parameter corresponds to columns, the fan-out parameter corresponds to rows in the array. Proper inclusion between adjacent entries is indicated by a separation line.

To conclude, we observe that the rank parameter can be traded with the fan-out parameter, as shown in the next result which is stated for class MCFG. In the proof, we will use the following convention: a sequence $X_i, \ldots, X_j$ denotes the empty sequence whenever $j = i - 1$.

Theorem 11 *Let $f \geq 1$ and $r \geq 3$. Then for $1 \leq k \leq r-2$ we have* $\mathcal{L}(r\text{-MCFG}(f)) \subseteq \mathcal{L}((r-k)\text{-MCFG}((k+1)f))$.

Proof. Let $G = (V_{\rm N}, V_{\rm T}, P, S)$ be a grammar in r-MCFG(f) and let k be an integer, $1 \leq k \leq r-2$. We will exhibit a grammar G' in $(r-k)$-MCFG($(k+1)f$) such that $L(G') = L(G)$. Let $G' = (V'_{\rm N}, V_{\rm T}, P', S)$, where

$$V'_{\rm N} = V_{\rm N} \cup \{A_{p,i} \mid p \in P \text{ and } 0 \leq i \leq k-1\}.$$

We define P' as follows. For each production $p \in P$, $p : A \to g(B_1, \ldots, B_t)$ with $t = \rho(p) > r-k$, we add the following productions to P':

$$\begin{aligned}
&p' : A \to g'(B_1, \ldots, B_{t-k-1}, A_{p,0}),\\
&p_0 : A_{p,0} \to g_0(B_{t-k}, A_{p,1}),\\
&p_1 : A_{p,1} \to g_1(B_{t-k+1}, A_{p,2}),\\
&\vdots\\
&p_{k-2} : A_{p,k-2} \to g_{k-2}(B_{t-2}, A_{p,k-1}),\\
&p_{k-1} : A_{p,k-1} \to g_{k-1}(B_{t-1}, B_t).
\end{aligned}$$

The functions g_i's introduced above simply form larger and larger tuples from their arguments, without appending any strings:

$$\begin{aligned}
&g_{k-1}(\langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-1})}\rangle, \langle x_{2,1}, \ldots, x_{2,\varphi(B_t)}\rangle)\\
&\qquad = \langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-1})}, x_{2,1}, \ldots, x_{2,\varphi(B_t)}\rangle,\\
&g_{k-2}(\langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-2})}\rangle, \langle x_{2,1}, \ldots, x_{2,\varphi(B_{t-1})+\varphi(B_t)}\rangle)\\
&\qquad = \langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-2})}, x_{2,1}, \ldots, x_{2,\varphi(B_{t-1})+\varphi(B_t)}\rangle,\\
&\vdots\\
&g_1(\langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-k+1})}\rangle, \langle x_{2,1}, \ldots, x_{2,\varphi(B_{t-k+2})+\cdots+\varphi(B_t)}\rangle)\\
&\qquad = \langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-k+1})}, x_{2,1}, \ldots, x_{2,\varphi(B_{t-k+2})+\cdots+\varphi(B_t)}\rangle,\\
&g_0(\langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-k})}\rangle, \langle x_{2,1}, \ldots, x_{2,\varphi(B_{t-k+1})+\cdots+\varphi(B_t)}\rangle)\\
&\qquad = \langle x_{1,1}, \ldots, x_{1,\varphi(B_{t-k})}, x_{2,1}, \ldots, x_{2,\varphi(B_{t-k+1})+\cdots+\varphi(B_t)}\rangle.
\end{aligned}$$

Thus, for $0 \leq i \leq k-1$, we have the following relation:

$$\varphi(g_i) = \sum_{h=0}^{k-i} \varphi(B_{t-h}) \quad \leq (k+1)f.$$

Now let us turn to function g' used in production p'. In order to define this function, we first introduce a homomorphism h from $\{x_{i,j} \mid 1 \leq i \leq t, 1 \leq j \leq \varphi(B_i)\} \cup V_{\rm T}$ to $\{x_{i,j} \mid 1 \leq i \leq t-k-1, 1 \leq j \leq \varphi(B_i)\}$

$\cup\{x_{t-k,j} \mid 1 \leq j \leq \varphi(B_{t-k}) + \cdots + \varphi(B_t\} \cup V_{\mathrm{T}}$. Homomorphism h is specified as follows:

$$h(\xi) = \begin{cases} a & \text{if } \xi = a,\ a \in V_{\mathrm{T}}; \\ x_{i,j} & \text{if } \xi = x_{i,j},\ 1 \leq i \leq t-k-1; \\ x_{t-k,\varphi(B_{t-k})+\cdots+\varphi(B_{i-1})+j} & \text{if } \xi = x_{i,j},\ t-k \leq i \leq t. \end{cases}$$

Now assume that g is defined by a relation of the kind:

$$g(\langle x_{1,1}, \ldots, x_{1,\varphi(B_1)}\rangle, \ldots, \langle x_{t,1}, \ldots, x_{1,\varphi(B_t)}\rangle) = \langle \alpha_1, \ldots, \alpha_{\varphi}(p)\rangle.$$

Then we have:

$$\begin{aligned} g'(&\langle x_{1,1}, \ldots, x_{1,\varphi(B_1)}\rangle, \ldots, \langle x_{t-k-1,1}, \ldots, x_{t-k-1,\varphi(B_{t-k-1})}\rangle, \\ &\langle x_{t-k,1}, \ldots, x_{t-k,\varphi(B_{t-k})+\cdots+\varphi(B_t)}\rangle) \\ &\quad = \langle h(\alpha_1), \ldots, h(\alpha_{\varphi(p)})\rangle \end{aligned}$$

Thus, $\rho(G') \leq r-k$, and $\varphi(G') \leq (k+1)f$. It can easily be seen that $L(G') = L(G)$. ■

We remark that, in the above theorem, the containment is proper, since we have already observed that there exist languages generated by 1-LUSCG$(f+1)$ that cannot be generated by class (-LUSCG(f)), for eaach $f \geq 1$. The above result transfers in the obvious way to the other parallel rewriting systems discussed in this section.

6 Remarks

We have characterized finite copying parallel rewriting systems by imposing a restriction, called locality, in the definition of derivation for the class USCG. This significantly alters the formal properties of USCG. While LUSCG is known to generate only semi-linear languages [Engelfriet *et al.*, 1980; Weir, 1988; Seki *et al.*, 1991], USCG can also generate non-semi-linear languages [Dassow and Păun, 1989]. And while the class $\mathcal{L}$(LUSCG) is only composed of languages in P, that is languages whose sentences can be recognized in deterministic polynomial time [Vijay-Shanker *et al.*, 1987], USCG can generate NP-complete languages [Dahlhaus and Warmuth, 1986]. As shown in this work, rewriting systems in LUSCG generate an infinite non-collapsing hierarchy with respect to the fan-out and rank parameters. The result implies that these rewriting systems do not admit normal forms

that are defined by some bound on both complexity measures. In contrast, two-normal forms are admitted for grammars in USCG, with respect to both parameters. This result has been shown for matrix grammars (see for instance [Dassow and Păun, 1989]) and it unproblematically transfers to USCG. Furthermore, it has been conjectured that language $L = \{ww | w \in D_1\}$ (see Section 2) is not in $\mathcal{L}$(USCG) [Dassow and Păun, 1989, p.42]; if this conjecture in fact holds, then $\mathcal{L}$(USCG) and $\mathcal{L}$(LUSCG) are incomparable.

The results of Section 3 have also interesting consequences for the recognition/parsing problem of the generated languages, as discussed in the following. Tabular methods for the solution of the recognition problem for the class MCFG have been presented in [Seki *et al.*, 1991], generalizing in this way the well known Cooke-Kasami-Younger tabular method for the recognition of context-free languages [Younger, 1967; Aho and Ullman, 1972]. It is worth observing that, in contrast with the Cooke-Kasami-Younger method, these methods do not behave "uniformly" on MCFG, in the sense that for each grammar $G \in$ MCFG a method using a recognition matrix with a number of dimensions proportional to d_G is needed, where $d_G = \varphi(G) \cdot (\rho(G)+1)$ is called the *degree* of G. The existence of a k-rank normal form G for any G' in the class MCFG, such that G can be obtained in polynomial deterministic time from G', would have entailed that the universal recognition problem for MCFG could be solved in deterministic polynomial time. This is quite unlikely, since in [Kaji *et al.*, 1992] and [Satta, 1992] NP-completeness results were independently shown for the recognition problem for classes MCFG(f), $f \geq 2$. (These results easily transfer to classes LUSCG(f).) Assuming P $\neq$ NP, the existence of a k-rank normal form of size exponential with respect to the size of the input grammar was still an open issue, leading to a possible solution to the uniform recognition of these languages. The result presented in this paper shows that tabular methods of the kind usually employed in context-free language recognition are not a viable solution to the problem.

A Appendix: Equivalence of LUSCG and MCFG

Class MCFG has been introduced in Definition 8 and an equivalence relation between MCFG and LUSCG has been stated in Theorem 6; this appendix provides the proof of Theorem 6. We have already remarked that the recursive definition of the rewrite relation in MCFG observes the local-

ity restriction. As a consequence, we find that in MCFG derivations can be associated with underlying trees that can be generated by context-free grammars. We develop here this idea and introduce concepts analogous to those presented in Definition 3.

For a given context-free grammar G_c, we call *complete* any derivation of the form $A \stackrel{*}{\Rightarrow}_{G_c} \eta$, η a string of terminals of G_c. As usual, we can represent derivations in G_c by means of trees whose nodes are labeled by symbols of G_c. We write $T(G_c)$ to denote the set of trees representing all complete derivations in G_c. Let $G = (V_{\rm N}, V_{\rm T}, P, S)$ be a multiple context-free grammar. Define $P^{(0)} = \{p \mid \rho(p) = 0\}$ and $P^{(1)} = P - P^{(0)}$. (We are overloading symbols $P^{(0)}$ and $P^{(1)}$; it will always be clear from the context whether these symbols denote subsets of productions of a grammar in LUSCG or of a grammar in MCFG.) Without loss of generality, we assume that p_S is the only production in P that rewrites S and $p_S \in P^{(1)}$.

Definition 9 *The* derivation grammar *of G, written* $\mathrm{der}(G)$*, is a context-free grammar $(P^{(1)}, P^{(0)}, \Pi, p_S)$, where $P^{(1)}$ and $P^{(0)}$ are the sets of nonterminal and terminal symbols respectively, p_S is the initial symbol and Π is a (finite) set of productions specified as follows. For every $p : A \rightarrow g(B_1, \ldots, B_{\rho(p)})$ in P and for every sequence $p_1, \ldots, p_{\rho(p)}$ of productions such that the left-hand side of p_i is B_i, $1 \leq i \leq \rho(p)$, production $p \rightarrow p_1 \cdots p_{\rho(p)}$ belongs to Π.*

It should be clear that any instance $A \Rightarrow_G \langle y_1, \ldots, y_{\varphi(A)} \rangle$ of the derivation relation in G can be associated with a derivation in $\mathrm{der}(G)$ of the form $p \stackrel{*}{\Rightarrow}_{\mathrm{der}(G)} \eta$ for some $p \in P$ and $\eta \in (P^{(0)})^*$, that is with a derivation tree in $T(\mathrm{der}(G))$ with root node labeled by p.

The main idea in the next theorem is to compare underlying context-free derivations in MCFG with underlying context-free derivations in LUSCG. To do so, we need to extend the rewrite relation in LUSCG to string tuples. Let $G = (V_{\rm N}, V_{\rm T}, P, S)$ be a grammar in LUSCG; in what follows we write $(\langle \gamma_1, \ldots, \gamma_n \rangle, I_1) \Rightarrow_G (\langle \delta_1, \ldots, \delta_n \rangle, I_2)$, $n \geq 1$, whenever $(\gamma_1 \cdots \gamma_n, I_1) \Rightarrow_G (\delta_1 \cdots \delta_n, I_2)$ holds. Let p be a production in G having left-hand tuple $(A_1, \ldots, A_n)$, $n \geq 1$. Similarly to the case of MCFG, any derivation in G having the form $(\langle A_1, \ldots, A_n \rangle, I^{\langle A_1 \cdots A_n \rangle}) \stackrel{*}{\Rightarrow}_G (\langle w_1, \ldots, w_n \rangle, I)$, $w_i \in V_{\rm T}^*$ for $1 \leq i \leq n$, can be associated with a complete derivation in $\mathrm{der}(G)$ of the form $p \stackrel{*}{\Rightarrow}_{\mathrm{der}(G)} \eta$, $\eta \in (P^{(0)})^*$, that is with a derivation tree in $T(\mathrm{der}(G))$ with root node labeled by p. We are now ready to prove Theorem 6, whose statement is repeated here.

Theorem 6 *Let r, f be integers such that $r, f \geq 1$. Then we have $\mathcal{L}(r\text{-MCFG}(f)) = \mathcal{L}(r\text{-LUSCG}(f))$.*

Proof. ($\subseteq$): Let $G = (V_{\rm N}, V_{\rm T}, P, S)$ be in r-MCFG(f). We construct G' in r-LUSCG(f) such that $L(G) = L(G')$. In what follows, let $p : A \rightarrow g(B_1, \ldots, B_{\rho(p)})$ be a production in P and let g be defined by an equation of the form

$$g(\langle x_{1,1}, \ldots, x_{1,\rho(B_1)}\rangle, \ldots, \langle x_{\rho(p),1}, \ldots, x_{\rho(p),\varphi(B_{\rho(p)})}\rangle) = \langle y_1, \ldots, y_{\varphi(A)}\rangle.$$

Assume also that symbol $p_{S'}$ does not denote any production in P. We define

$$V'_{\rm N} = \{[p, i, j] \mid p \in P, 1 \leq i \leq \rho(p), 1 \leq j \leq \varphi(B_i)\} \cup \{[p_{S'}, 0, 0]\}$$

and $G' = (V'_{\rm N}, V_{\rm T}, P', [p_{S'}, 0, 0])$, where set P' is constructed as follows. We associate with each p, specified as above, a homomorphisms h_p mapping set $\{x_{i,j} \mid 1 \leq i \leq \rho(p),\ 1 \leq j \leq \varphi(B_i)\} \cup V_{\rm T}$ into set $V'_{\rm N} \cup V_{\rm T}$ and defined as follows:

$$h_p(\xi) = \begin{cases} [p, i, j] & \text{if } \xi = x_{i,j}; \\ \xi & \text{if } \xi \in V_{\rm T}. \end{cases}$$

Assume that p' is a production in P containing in the k-th position of its right-hand side the symbol in the left-hand side of p. We add to P' the production

$$([p', k, 1], \ldots, [p', k, \varphi(A)]) \rightarrow (h_p(y_1), \ldots, h_p(y_{\varphi(A)})).$$

We iterate the process for every pair p, p' as above. In addition, let $p_S : S \rightarrow g(B_1, \ldots, B_{\rho(p_S)})$ be defined by an equation (recall that $\varphi(p_S) = 1$)

$$g(\langle x_{1,1}, \ldots, x_{1,\varphi(B_1)}\rangle, \ldots, \langle x_{\rho(p_S),1}, \ldots, x_{\rho(p_S),\varphi(B_{\rho(p_S)})}\rangle) = \langle y_1\rangle.$$

We add to P' the production

$$([p_{S'}, 0, 0]) \rightarrow (h_p(y_1)).$$

Note that $\rho(G) = \rho(G')$ and $\varphi(G) = \varphi(G')$.

We claim that $A \Rightarrow_G \langle y_1, \ldots, y_{\varphi(A)}\rangle$ if and only if there exists a derivation in G' of the form

$$(\langle [p', k, 1], \ldots, [p', k, \varphi(A)]\rangle, I^{\langle [p',k,1],\ldots,[p',k,\varphi(A)]\rangle}) \stackrel{*}{\Rightarrow}_{G'} (\langle y_1, \ldots, y_{\varphi(A)}\rangle, I),$$

for some p' and k as in the definition of $V'_{\rm N}$, and for some I. The claim can be easily established by associating with the derivations above the corresponding trees in $T(\mathrm{der}(G))$ and $T(\mathrm{der}(G'))$ (which have the same height) and then proceeding by induction on the height of these trees. Relation $L(G) = L(G')$ immediately follows from the claim.

($\supseteq$): Let $G = (V_{\rm N}, V_{\rm T}, P, S)$ be in r-LUSCG(f) and let $\mathrm{der}(G) = (P^{(1)}, P^{(0)}, \Pi, p_S)$ be the derivation grammar associated with G. We construct $G' \in r$-MCFG(f) such that $L(G) = L(G')$. Define

$$V'_{\rm N} = \{[A_1, \ldots, A_{\varphi(p)}] \mid (A_1, \ldots, A_{\varphi(p)}) \text{ is the left-hand tuple of } p \in P\}$$

and let $G' = (V'_{\rm N}, V_{\rm T}, P', [S])$. Let p_0 be a production in P of the form

$$p_0 : (A_1, \ldots, A_{\varphi(p_0)}) \to (\alpha_1, \ldots, \alpha_{\varphi(p_0)})$$

and let p be a production in Π of the form $p_0 \to p_1 \cdots p_n$. Let also $(B_{i,1}, \ldots, B_{i,\varphi(p_i)})$ be the left-hand tuple of p_i, $1 \le i \le n$. Consider a complete rewriting in G of the right-hand tuple of p_0 by means of productions p_i, $1 \le i \le n$; call σ such a rewriting. For $1 \le h \le \varphi(p_0)$, construct σ_h from α_h and σ by replacing nonterminal $B_{i,j}$ in α_h with $x_{i,j}$ if the j-th context-free production composing scattered context production p_i is used to rewrite $B_{i,j}$ in σ. Then we add to P' the production

$$[A_1, \ldots, A_{\varphi(p_0)}] \to g_\sigma([B_{1,1}, \ldots, B_{1,\varphi(p_1)}], \ldots, [B_{n,1}, \ldots, B_{n,\varphi(p_n)}]),$$

where g_σ is a linear regular function of arity $\varphi(p_0)$ and rank n defined by equation

$$g_\sigma(\langle x_{1,1}, \ldots, x_{1,\varphi(p_1)}\rangle, \ldots, \langle x_{n,1}, \ldots, x_{n,\varphi(p_n)}\rangle) = \langle \sigma_1, \ldots, \sigma_{\varphi(p_0)}\rangle.$$

In the construction of P', this process is iterated for every p_0, p and σ as above. Note that $\rho(G) = \rho(G')$ and $\varphi(G) = \varphi(G')$.

We claim that $(\langle A_1, \ldots, A_{\varphi(A)}\rangle, I^{\langle A_1, \ldots, A_{\varphi(A)}\rangle}) \stackrel{*}{\Rightarrow}_G (\langle y_1, \ldots, y_{\varphi(A)}\rangle, I)$ for some I, if and only if there exists a derivation in G' of the form $[A_1, \ldots, A_{\varphi(A)}] \stackrel{*}{\Rightarrow}_{G'} \langle y_1, \ldots, y_{\varphi(A)}\rangle$. As before, this can be easily established by induction on the common height of trees in $T(\mathrm{der}(G))$ and $T(\mathrm{der}(G'))$ associated with the above derivations. Again, relation $L(G) = L(G')$ immediately follows from the claim. ■

References

[Aho and Ullman, 1969] A. V. Aho and J. D. Ullman. Properties of syntax directed translations. *Journal of Computer and System Science*, 3(3):319–334, 1969.

[Aho and Ullman, 1971] A. V. Aho and J. D. Ullman. Translations on a context-free grammar. *Information and Control*, 19:439–475, 1971.

[Aho and Ullman, 1972] A. V. Aho and J. D. Ullman. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ, 1972.

[Bauderon and Courcelle, 1987] M. Bauderon and B. Courcelle. Graph expressions and graph rewritings. *Mathematical Systems Theory*, 20:83–127, 1987.

[Dahlhaus and Warmuth, 1986] E. Dahlhaus and M. K. Warmuth. Membership for growing context-sensitive grammars is polynomial. *Journal of Computer and System Science*, 33:456–472, 1986.

[Dassow and Păun, 1989] J. Dassow and G. Păun. *Regulated Rewriting in Formal Language Theory*. Springer Verlag, Berlin, Heidelberg, New York, 1989.

[Engelfriet and Heyker, 1991] J. Engelfriet and L. Heyker. The string generating power of context-free hypergraph grammars. *Journal of Computer and System Science*, 43:328–360, 1991.

[Engelfriet *et al.*, 1980] J. Engelfriet, G. Rozenberg, and G. Slutzki. Tree transducers, L systems, and two-way machines. *Journal of Computer and System Science*, 20:150–202, 1980.

[Greibach and Hopcroft, 1969] S. Greibach and J. Hopcroft. Scattered context grammars. *Journal of Computer and System Science*, 3:233–247, 1969.

[Habel and Kreowsky, 1987] A. Habel and H. J. Kreowsky. Some structural aspects of hypergraph languages generated by hyperedge replacement. In *Proceedings of STACS*, pages 207–219, Berlin, Germany, 1987. Springer-Verlag. Lecture Notes in Computer Science, volume 247.

[Hopcroft and Ullman, 1979] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, MA, 1979.

[Joshi *et al.*, 1991] A. Joshi, K. Vijay-Shanker, and D. Weir. The convergence of mildly context-sensitive grammatical formalisms. In P. Sells, S. Shieber, and T. Wasow, editors, *Foundational Issues in Natural Language Processing.* MIT Press, Cambridge MA, 1991.

[Kaji *et al.*, 1992] Y. Kaji, R. Nakanisi, H. Seki, and T. Kasami. The universal recognition problems for multiple context-free grammars and for linear context-free rewriting systems. *IEICE Transactions on Information and Systems*, E75-D(1):78–88, 1992.

[Kasami *et al.*, 1987] T. Kasami, H. Seki, and M. Fujii. Generalized context-free grammars, multiple context-free grammars and head grammars. Technical report, Osaka University, 1987.

[Latteux, 1979] M. Latteux. Substitutions dans les EDT0L-systèmes ultralinéaires. *Information and Control*, 42:194–260, 1979.

[Mayer, 1972] O. Mayer. Some restrictive devices for context-free grammars. *Information and Control*, 20:69–92, 1972.

[Milgram and Rosenfeld, 1971] D. Milgram and A. Rosenfeld. A note on scattered context grammars. *Information Processing Letters*, 1:47–50, 1971.

[Rozenberg and Vermeir, 1978] G. Rozenberg and D. Vermeir. On ET0L systems of finite index. *Information and Control*, 38:103–133, 1978.

[Rozenberg, 1973] G. Rozenberg. Extension of tabled 0L-system and languages. *International Journal of Computer and Information Science*, 2:311–336, 1973.

[Salomaa, 1973] Arto Salomaa. *Formal Languages.* Academic Press, Orlando, Forida, 1973.

[Satta, 1992] Giorgio Satta. Recognition of linear context-free rewriting systems. In *30th Meeting of the Association for Computational Linguistics (ACL'92)*, 1992.

[Seki *et al.*, 1991] H. Seki, T. Matsumura, M. Fujii, and T. Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229, 1991.

[Thatcher, 1973] J. W. Thatcher. Tree automata: An informal survey. In A. V. Aho, editor, *Currents in the Theory of Computing*, chapter 4, pages 143–172. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[Vijay-Shanker *et al.*, 1987] K. Vijay-Shanker, D. J. Weir, and A. K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In 25^{th} *Meeting of the Association for Computational Linguistics (ACL'87)*, 1987.

[Weir, 1988] D. J. Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1988.

[Weir, 1992] D. J. Weir. Linear context-free rewriting systems and deterministic tree-walk transducers. In *Proc. of the 30^{th} Meeting of the Association for Computational Linguistics (ACL'92)*, Newark, Delaware, 1992.

[Younger, 1967] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208, 1967.